Network Working Group
Request for Comments 184
NIC 7128
Category: D.6

Karl Kelley
University of Illinois
6 July 1971

# Proposed Graphic Display Modes

The ARPA Network node at the University of Illinois' Center for Advanced Computation is somewhat different from other nodes in that we are not simply attaching an existing computer center to the net. We are in the process of establishing the computer system specifically for use of the ILLIAC IV and the Network. In this mode we are establishing operating systems, network interface and utility routines, and ILLIAC IV routines to be used over the network.

In the field of computer graphics we are in the process of building a system essentially from scratch. The building blocks of this capability comprise a small -- but growing -- collection of display hardware and a small cadre of persons with experience on separate and unique graphics equipment at the University of Illinois. Starting as we are with little-or-no system type software for computer graphics, we have a once-only opportunity to provide the system with computer graphics applications and utility programs which encompass all the features and capabilities that have heretofore been available only in bits and pieces at various separate installations.

It is apparent at the outset that the design for this system will be heavily weighted toward a network-type usage. For this reason we are eager to ensure that our system data structures, files, etc., be as nearly compatible the Network Graphics Protocol as is practicable. Our initial planning and first-version system will be pointed at the network type of operation and we hope to stay flexible enough to employ the Network Protocol on a local basis (between our PDP-11 and the B6500) as the protocol is developed.

We have been considering (in the planning of our system and pondering the protocol problem) just what display modes we would want to have available and thus would want the protocol to include. The purpose of this RFC is to outline our initial thoughts on the matter and to interact with other nodes about how they can/should be included in the protocol. We intend here not to belabor display modes which are certain to be needed everywhere, such as vectors, points, and characters, but rather to summarize those and outline in more detail only those which are slightly different.

The display system, and the network protocol, will require something like the following list of display types:

1. Points — Including normal points, plot a symbol at a point, plot a point with intensity
2. Lines(two-point)
3. Vectors (from present beam location to a point)

(2 and 3 include visible and not visible, dotted, dashed, overbright, and ?)

4. Character Streams
5. Viewport and Window Specifications (ala LDS-1)
6. Transformations (scaling and rotation) of Instances
7. Equipment-Specific Byte Streams
8. Read-Back of Keyboards, Function Buttons, Cursors, Etc.

It seems clear that some type of list or ring structure will be needed to handle our display files. Whether that structure need be a part of the protocol is not evident (after all, you could just send the equipment byte-stream), but it is our feeling that it will be needed. It is evident that the protocol must anticipate the needs of various popular display devices as CalComp, Computek, and similar storage displays, interactive displays such as Adage, LDS-1, Vector General, IDIOM, grey-level displays like the PEP-1 and raster-oriented gadgets like the Gould, Versatec, and garden-variety line printers.

The standard display element types given above are assumed to be in common use. A point, for example, is a pen-down command followed by a pen-up command on a plotter. On an interactive device it is merely an intensification of the beam. To plot a symbol at a point the pen (beam) is moved to the point with pen up (beam off) and then the symbol is plotted incrementally with the pen down (beam on). Graphics devices with beam intensity control will be expected to handle the "plot-point-with-intensity" format.

In terms of the standard display types the only difference between lines of the two-point form and vectors is that in the former, four data items are needed for each segment while in the latter, only two data items are needed. In either case, the user should be able to reposition the pen (beam) absolutely by drawing an invisible line. He should also be able to plot dotted or dashed lines without having to separately specify each point or short vector. In instances where emphasis is needed it is useful to be able to selectively intensify a line, or in the case of interactive displays, make it blink.

Character streams are used either for text or for labeling drawings. In most applications the character stream is specified by its starting location in screen coordinates, the number of characters, and the location of a buffer of characters to be used. In some purely output graphics systems the character stream is specified via a format similar to printer output, with the characters being placed in the specified area on the display.

Items 5 and 6 of the above list primarily apply to displays like Sketchpad and the Evans and Sutherland display. Viewport is taken to be synonymous with "observer parameters in the object space" while a window means "selected portion of the display surface". Transformations of instances refers to a feature of Sketchpad which allows multiple uses of the same "pattern" for a graphical element.

Because new equipment is constantly coming into use, and special-purpose equipment is available at some nodes, it is prudent to have available a capability for sending equipment-specific information over the net as a part of the protocol. Such information would be in the form of byte streams formatted according to the equipment specifications and pointed at the proper node and equipment. It would not be expected that each node be able to interpret the nonstandard byte stream. Also very equipment specific is the information passed from an interactive device back to the originating program. Elements such as joystick or cursor position, lightpen hits, function buttons pressed, etc., are inherently dependent upon the device employed. Although these devices are widely used, their general dependence upon display buffers, display lists, or interrupts is of special concern to the network graphics protocol.

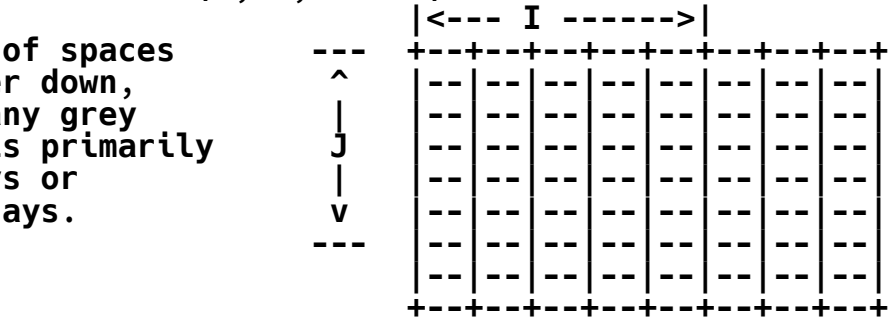
We are interested in expanding upon the uses of grey-scale display modes for representation of computer-generated data and for three-dimensional object representation. To facilitate this, our system will have available at least four, program-callable display element types for production of pictures on grey-level display devices. The initial names for these modes are the procedure names: GRIDAREA, MATRIXAREA, SCANLINE, and SCANPOINT. The following paragraphs will outline how the modes operate both from a user and a data-

communications point of view. The specific hardware involved is not specified, nor do we ignore the possibility that hardware can be designed to operate this way directly. For example, the SCANLINE is precisely the way one display does operate. In the interim, however, software sits between these procedures and actual devices.

GRIDAREA
--------
The user specifies, in an initial call, the size grid he wishes to use and the number of intensity levels he will need. Subsequent calls send, as data items, the i, j locations of an area and a byte for intensity.

The initializing call is GRIDSET(N, M, IRNGE)

where N is the number of spaces across, M is the number down, and IRNGE tells how many grey levels to use. This is primarily for grey-scale displays or pseudogrey-scale displays.




On a Gould Electrostatic Printer-Plotter, for example, the system would expect to have dot patterns with which to fill areas in order to simulate the greys. For purposes of other displays, a SETSQUARES procedure should be available so that the user can specify various kinds of cross- hatching and character filling to apply to the grid areas.

To enter an item of data the procedure is

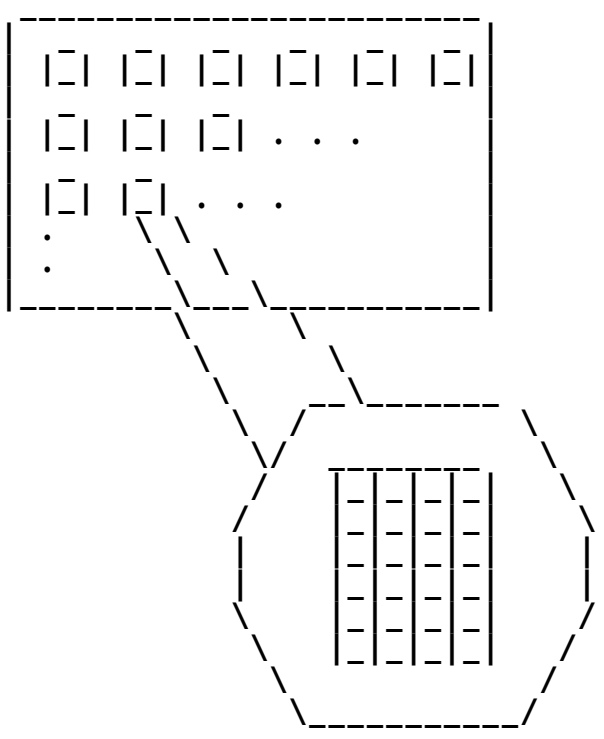
```
                                    1 <= I <= N
          GRIDAREA(I, J, LEVEL)     1 <= J <= M
                                    1 <= LEVEL <= IRNGE
```

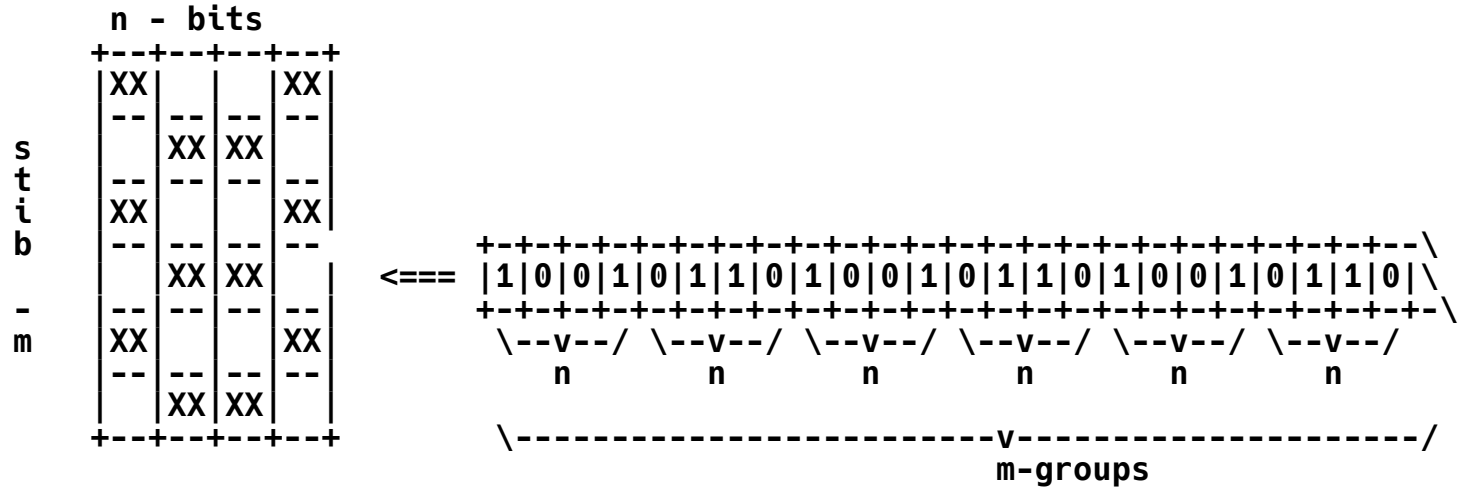
where I and J select an area on the grid, and LEVEL tells how to fill it. Obviously, for this kind of display mode some provision must be made to end the picture because the servicing routine will have to work on it from either the top or the bottom in a sweep mode. A procedure call of GRIDAREA(0, 0, 0) will terminate it. The GRIDAREA mode is very similar to the following, MATRIXAREA, with the primary difference being that the specification of areas is random and areas which are not specified will be left blank.

**MATRIXAREA** — The user specifies an area size and a "pseudo-character" set, then writes from left to right, top to bottom, much like a line printer, using bytes to specify which of his "pseudo-character" set to use.

The initializing call is MATRIXSET(N, M, DEFINITION, CODE)

The display mode is raster oriented, and each "pseudo-character" will be on a N x M matrix of dots. (A later embellishment for printers would include matrix of characters.) Parameters DEFINITION and CODE are both arrays, used together to specify the "pseudo-character" set. DEFINITION is packed with bits according to the following scheme:





The bit stream for the definition is irrespective of word boundaries. We leave it up to the MATRIXSET routine to be able to undo this. Obviously, for user convenience we have some standard sets they can use to save having to define their own, and if they want to define their own, we make routines to ease the pain.

The array CODE, on the other hand, is a byte-stream, i.e., a stream of eight-bit groups of bits which will correspond to each of the groups of (N x M)-bits. This allows 256 pseudocharacters for one set.

To enter an item of data in this mode the procedure is

MATRIXAREA(ARRAY, LENGTH)

ARRAY is a buffer location and length is the number of code bytes which are to be put out. Each call will put out one row of the display. Unused bytes at the end of the stream (bytes left over in the last word) should be zero. Any codes in excess of the maximum number allowed on a line will be discarded.

It should be noted that this routine is nominally used for special character sets, or for that matter, any character sets that are software generated on dot-raster devices. In addition, however, it can be used for photomosaic displays, area filling on maps, and development of capability of producing audiovisual aids for presentations.

| | |
|---|---|
| SCANLINE<br>-------- | A raster-scanning display mode for which the user specifies a raster size, number of grey levels needed, and direction of scan. The subsequent data items specify only the location and character of a change in the scanning beam (or program). |

The initializing procedure is SCANLINESET(DELTA, LEVELS, ORGMODE)

DELTA is an integer specifying the number of display points to be included in each step. LEVELS denotes the number of intensity levels to be used, and the sign of ORGMODE specifies whether the scan is to be from the bottom up (plus) or top down (minus) on the display. For both cases we will assume left-to-right. The absolute value of ORGMODE gives the starting Y position.

All subsequent calls to this routine are of the form

SCANLINE(X, INTENSITY)

The first such call denotes the origin in X and the initial intensity. Subsequent calls denote the X value of the next point on the scan where the intensity is to change, and that new intensity. The program (or device) takes care of the stepping of the scan by DELTA across the page, with the current intensity. Thus, the program (device) only needs a data item for each change in the scan, not for each position. When the next X is less than the previous X, or the X position has been stepped to its limit, the Y position in the incremented or decremented to continue the scan on the next line.

We see the device which accepts such a display as accepting a stream of triplets of bytes, where the first two bytes (16 bits) specify the X and the third (8 bits) specifies the level. The end of the stream would be specified by three bytes of deletes (all ones). This display mode is implemented in hardware on the display at the Coordinated Science Laboratory at this University. It is the same one which was used for the grey-scale work which has been reported by Bouknight.

| | |
|---|---|
| SCANPOINT<br>--------- | A point with intensity scanning mode in which the scan is handled automatically and only the intensity of each point needs to be transmitted to the program (device). |

The initializing call for this procedure is

SCANPOINTSET(DELTA, LEVELS, ORGMODE)

The arguments are the same as for SCANLINE. The difference is in the meaning of subsequent calls. The origin for the scan is at the left end of the line corresponding to the absolute value of ORGMODE. The stepping is done from left to right and at the end of each line, the Y position is incremented or decremented by DELTA, according to the sign of ORGMODE.

Subsequent calls to this procedure are of the form SCANPOINT(LEVEL) where LEVEL denotes the intensity level at which the next point is to be displayed. In this mode every point must have its intensity specified by a separate call to the routine (byte to the device). However, beyond the starting point no position information is required.