The economic advantages of printing internal-use documents on a raster printer have usually been limited to purely numeric and text documents. This paper describes an experimental character-graphic art program that demonstrates the potential of the IBM 3800 for printing a restricted set of character-graphic art documents. A special character set is outlined, as well as an algorithm that selects those characters from the set that best approximate any straight line. This character-graphic algorithm permits line art to be included in formatted text documents. There is no manual artwork or paste-up in the document output. The artwork for this paper has been reproduced from material printed by the technique discussed, although the body text has been reset from the 3800 output.

Experimental page makeup of text with graphics on a raster printer

by B. J. Shepherd

Publications for use within a company frequently contain text and graphics (line drawings, graphs, images). Some documents, such as maintenance manuals, typically consist of 30 percent high quality artwork. Other documents, such as procedures manuals, contain flow charts and similar simple diagrams. Line printers such as the IBM 1403 have been used to print such output by using appropriate available character sets. Similarly, the format character set of the IBM 3800 raster printer[1,2] can be used for simple drawings.

This paper describes an experimental effort aimed at the uncomplicated end of the art and document spectrum. The goal has been to study the potential of the 3800 to produce simple line art intermixed with text. Although few readers question the appeal or value of illustrations, there are several additional objectives. The artwork must be easy to define and modify. It should be visually pleasing (i.e. have adequate line quality). Finally, it is valuable to treat the text and artwork together, in order to be able to store the entire document digitally and eliminate expense and potential errors of hand collating figures into text after printing.

Once text and illustrations can be printed simultaneously with sufficient quality, copies of the document can be printed on demand. This eliminates the expense of shelf storage of extra copies, as well as keeping the stored copies current. The ability to print on demand means that active documents are more current. Since text and picture data are stored in a common digital data base, selective (custom) publication is possible. That is, each copy is not only a first copy, but it is also an edited and selected copy that contains only the information relevant to the recipient.

Characteristics of the 3800 raster printer

To understand the printing of simple artwork on the 3800, we examine those characteristics of the hardware and system software that bear on the production of line art.[3] Perhaps most important is the ability to load new type fonts (character descriptions) into the writable character generator memory. Characters are defined as a series of dots or print points. Output can be printed at 6, 8, or 12 lines per inch, and 10, 12, or 15 characters per inch.

The resolution is not symmetric. At 8 lines per inch and 12 characters per inch, the character cell or box is 15 print points wide and 18 print points high. The character boxes are contiguous, so both the interline spacing (leading) and intercharacter spacing must be included within the (15×18) character box. There are some additional limitations. For example, there is no generalized overstrike capability, although an equivalent compound character image can be defined. The page to be printed is treated as a mapped entity. This means, for a given character box size (given pitch and lines per inch), that characters can be placed only within the uniform array of rows and columns available on a conventional line printer.

Another limitation is the number of character codes available for use on a single printed page. The upper limit is 255, which includes the space/blank character for each pitch present. Character definitions can be loaded only during the time available between pages. One font (e.g., 12 pitch Gothic) requires 64 positions of character generator memory. If lower case and/or italics and/or boldface are added, additional character generator storage is required. Thus, the maximum number of character codes that can be reserved for use in generating lines is arbitrarily chosen to be 64 for this implementation.

Experimental graphic character set

A special set of graphic characters had to be defined. The members of this graphic character set can be used to approximate

straight or curved lines. Where two line segments join in a single character box, a new compound character incorporating both segments has been defined. There are two problems that must be solved in any character graphics scheme. First, an appropriate balance must be found between character set size (number of codes) and line drawing quality. Since the number of graphic codes has been set at 64, the problem reduces to one of maximizing the line drawing quality.

The second problem became apparent as soon as proof pages using the character set had been prepared, namely, that of selecting the approximating character codes manually. For other than vertical, horizontal, or diagonal lines at plus or minus 30, 45, or 60 degrees, it would usually take several tries to achieve the desired result. An escape from manual selection has been to develop an approximating algorithm to select characters from the character set. This algorithm is the subject of the next section.

Character graphic algorithm

character set

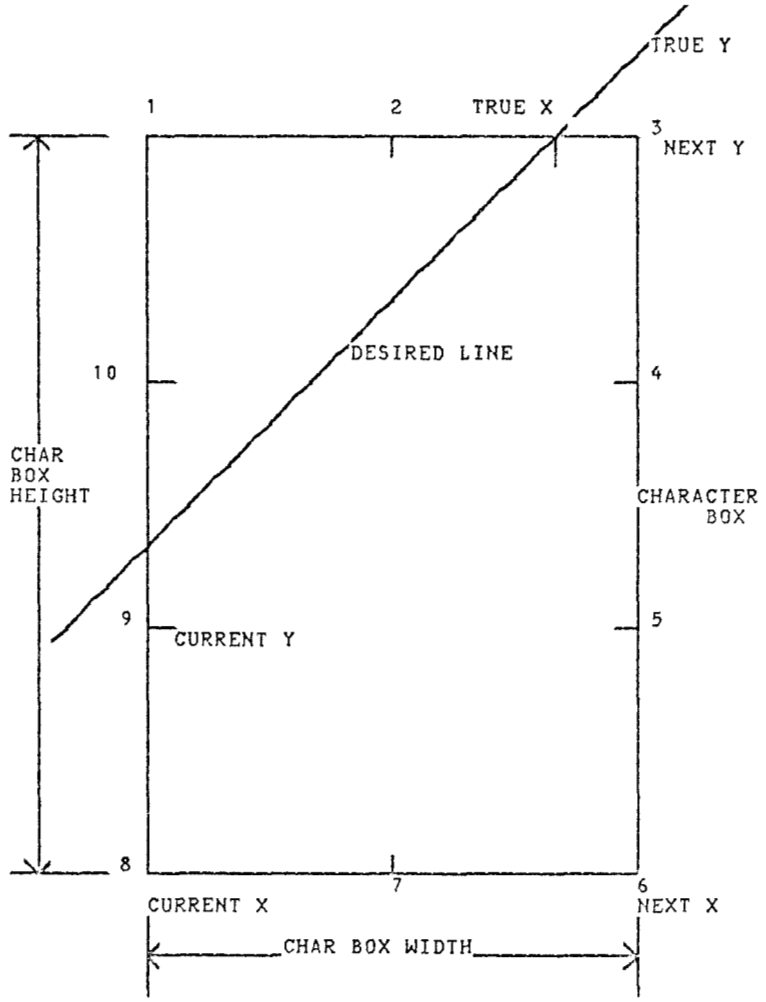
At the lowest level, the algorithm depends on the characteristics of the graphic character set. As indicated previously, the graphic character set is limited to 64 characters. This permits text and line art to be printed on the same page since 192 codes are reserved for alphabetic and other characters. The character set is defined for 8 lines per inch and 12 characters per inch to provide compatibility with the common preference for text character spacing. Ten numbered end points in the character box in Figure 1 are uniformly spaced around the edges of the character box or cell.

Lines are defined between all possible pairs of end points. Since direction does not matter, there are one-half the theoretically possible number of segments, that is, $10 \times 9/2 = 45$ line segments. Also, line segments between points 1, 2, and 3 are really in the next higher character box, and line segments between points 3, 4, 5, and 6 lie in the next character box to the right. There are thus 36 unique line segments within a character box.

The remaining 28 codes (to the limit of 64) are used for multiple-segment combination characters. The most common combination characters are corners (Ls) and junctions (Ts) of two lines. Since illustrations generally mimic real objects and most real objects are opaque, there are essentially no occurrences of line crossings.

The theoretical character set just described has one problem for real-world use. The line segments are infinitesimally thin and are thus invisible. When finite-width line segments are employed, a potential limitation on line quality becomes apparent. With some effort, it is possible to define a character set with no discontinui-

Figure 1 Definitions for approximation algorithm



ties between segments in vertically adjacent or horizontally adjacent character locations. For line segments in diagonally adjacent character boxes, an essential discontinuity appears due to the finite line width. This can be compensated using an auxiliary fillet character in an adjacent character box. In fact, four fillet characters are defined for steep and shallow lines of positive and negative slope.

Algorithm logic

The algorithm can be more easily understood in terms of the items defined in Figure 1, which is a reproduction of output produced on the 3800. The figure shows a character cell traversed from left to right by a positive slope line. The current Y is the

endpoint of the immediately prior approximating character segment. The slope and distance from the starting (initial) X to the next X are used to calculate the true Y. If the true Y value does not exceed the upper Y value (top of the character box containing the current Y), the true Y is rounded to pass through point 3, 4, 5, or 6 to generate the next Y. The current Y and next Y determine fully the character code that best approximates the line in the current character box.

For character segments that pass through point 3 or 6, a fillet character is specified for the character location to the right. The particular fillet code depends on the code of the line-approximating segment. If, in the preceding paragraph, the true Y value exceeds the upper Y value, a supplemental calculation is needed. The inverse slope and distance from the starting point are used to calculate the true X. The true X value is then rounded to point 1, 2, or 3 as the end point of the current line segment. Once again, the starting and ending points are sufficient to determine the segment code.

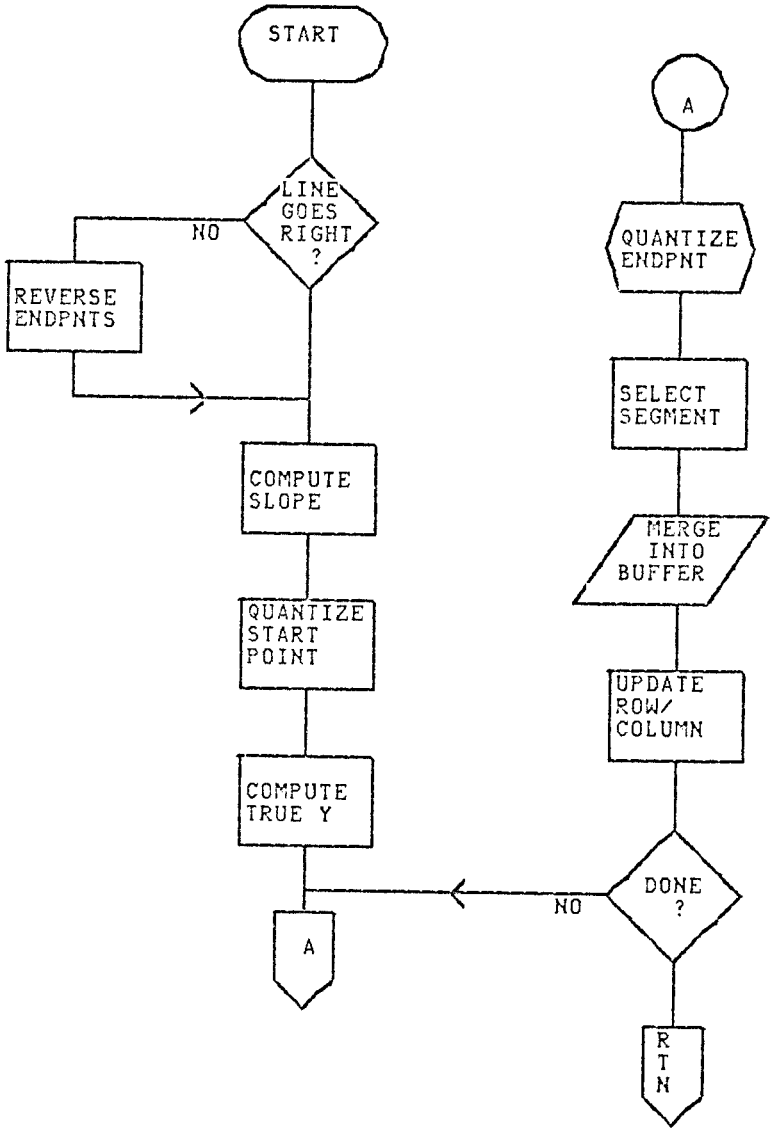
Now consider the general flow of the algorithm presented in Figure 2, which is also a reproduction of output produced on the 3800. If necessary, the starting and ending coordinates of the line are interchanged so that all lines are drawn to the right. This reduces the size of the algorithm by a factor of two. With suitable changes to Figure 2 and its description above, the algorithm can presume that all lines are going up, or down, or to the left. After the slope has been calculated and control variables initialized, the starting point of the first character (line segment) must be determined.

The origin of the character box is the lower left corner. If, upon rounding, the starting point of the line is not at the lower left corner of the character box, a fractional X and/or fractional Y is stored. The fractional X/Y data must be included in the calculation of the true fractional X/Y discussed in conjunction with Figure 1. Once the starting point has been determined, successive segments are determined as previously described until the right end of the line segment is reached. The procedure for negative slope lines is essentially the same. The merging of character codes (line segments) into the buffer is discussed in the next section.

Additional approximation algorithms for curves are presented in References 4–13. Reference 14 deals with curve generation for raster (rather than character) devices. Reference 15 discusses the case of a noncartesian coordinate system.

The algorithm is similar to the Bresenham algorithm.[16,17] There are, however, several differences, since the 3800 graphic charac-

Figure 2 General logic of selection algorithm for optimum approximating character



ter set emulates a plotter that can move in not just 4, or even 8, but rather 20 different directions. Also, the length of the step-moves is a function of their direction. That is, the plotter "unit square" is a rectangle. Finally, unlike a plotter, the 3800 does not provide physical overstrike, so software emulation must be provided.

Merging text and graphics

It was pointed out previously that the 3800 is not capable of arbitrary physical overstrikes as performed by an impact printer. For

that reason, special processing is needed to store the line segment character codes developed by the algorithm. To achieve the appearance of overstrikes, as is required to create the joining of two lines (segments) of different slopes, the entire page to be printed must be held in storage until all lines (and text) have been processed.

The processing of the page is straightforward. If the current contents of the page buffer location that is to receive the character segment just computed is null (blank), the segment is stored. If the buffer location contains a text character, the line segment does or does not replace the character, depending on the setting of a global mode control variable. Normally, text is not replaced, since geometric lines have even higher redundancy than text.

If the target buffer location already contains a line segment character, the existing and just-computed codes are used to determine whether a compound character containing both segments has been defined. If a compound character is defined for the two segments, it is placed in the buffer. If the needed compound character does not exist, the buffer contents are not altered. In addition to the compound characters, a short vertical or horizontal segment (partial character) is replaced by a collinear longer segment that is generated for the same location.

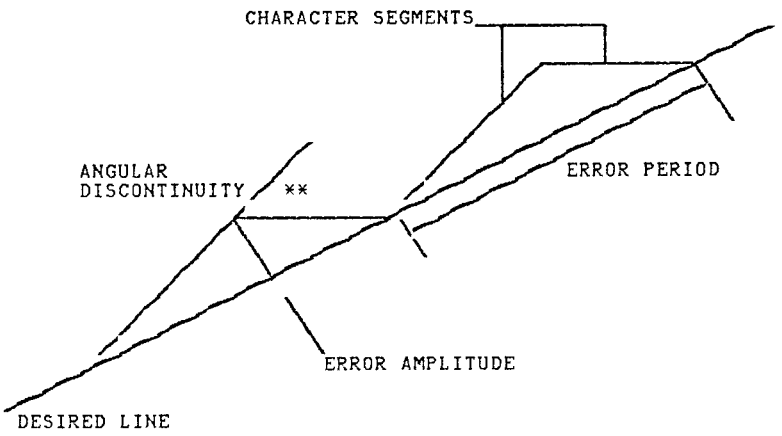
Line quality

One way to improve visual quality is to use fillet characters. There are, however, other ways to characterize the fidelity with which lines are reproduced by means of character graphics. Two qualitative measures are the number of slopes that can be plotted exactly and the end-point quantization measure. The character set presented here can reproduce vertical and horizontal lines as well as positive and negative slopes of 1/2, 1, 3/2, and 3. End points are quantized to a grid of points 1/24 inch apart both vertically and horizontally. It should be noted that the grid points in the interior of a character box are not allowed as line end points.

Although these characterizations are indicative, a more analytic measure of line quality is desirable. Such an analytic measure permits comparison and evaluation of several character-graphic character sets without the cost of full implementation. Thus the best character set can be chosen with minimum expense. Also, analytic tools can provide more consistent results than human factors experiments.

Figure 3, which is a copy of output produced by the method discussed in this paper, shows some quantities that can be used to develop analytic measures of line approximation fidelity. In addi-

Figure 3 Quantities used to describe line approximation fidelity



tion to *error amplitude, error period*, and *angular discontinuity*, it is useful to define *error power* as the product of error period and error amplitude. The measures are averaged over a set of standard lines, which may be nonuniformly weighted, for each of the character sets to be compared. Obviously other factors must be included in finally selecting a character set, such as the number (count) of characters needed and the suitability of implementing the algorithm in hardware or firmware.

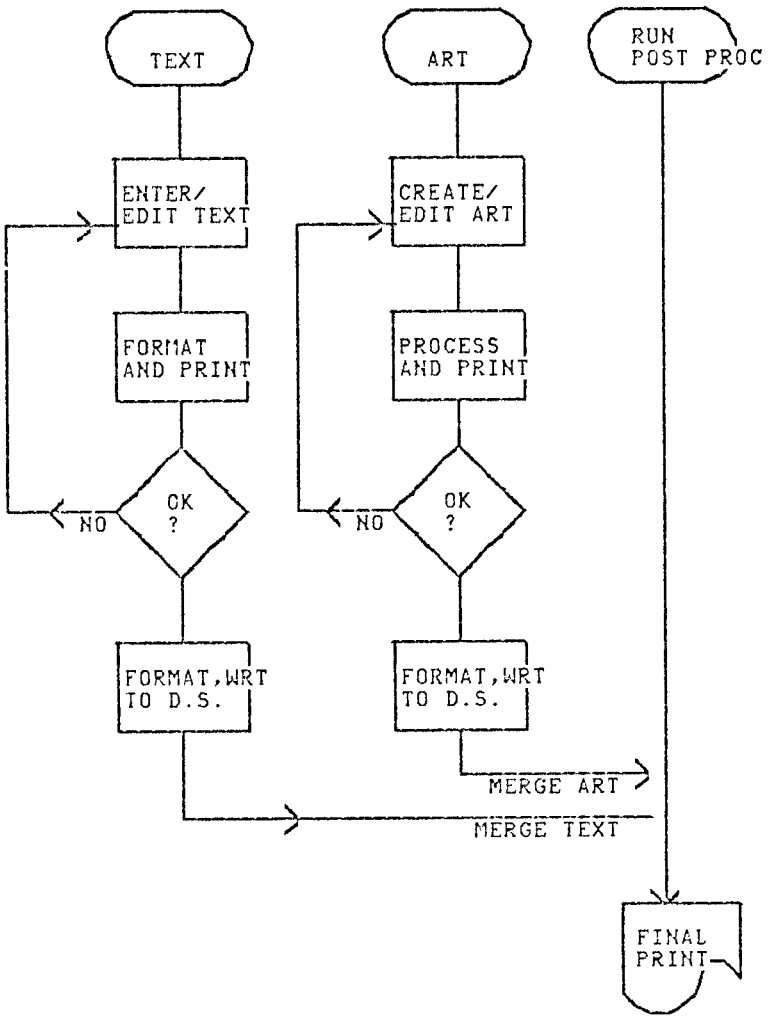
For the character set reported here, 10 lines in the first quadrant have been selected. The slopes are 0, 1/4, 1/3, 1/2, 3/4, 1, 3/2, 2, 3, and 9/2. The average error amplitude (over the 10 lines) is 3.9 percent of the error period. The error frequency is 2.1 cycles per inch, which corresponds to an error period of 0.48 inches. The average angular discontinuity is 10.5 degrees. Finally, the error power ($\times 100$) is 0.85. Each of these measures is a factor of three better than those for an earlier character set proposal.

Producing a document

A test document was entered and edited on a text editor, and the artwork was created and edited in successive batch runs, using list language commands. The text document was formatted, with space left for the illustrations, and the art was then correctly placed on each page by measuring rows and columns. The art for each page was defined as a subroutine named with the page number.

The final production run involved reading the art and storing each page of art as a subroutine. Then successive pages of formatted

Figure 4 Logical flow of production of test report



text were read, and the art was merged. The printed output, shown in Figure 4, was produced on an IBM 3800.

Concluding remarks

Line art can be produced on the IBM 3800, as demonstrated by the figures in this paper, which contain no hand-drawn art. A character set that produces good line quality has been described, together with an algorithm that is matched to it. The relationship between the characteristics of the 3800 and character graphics has been discussed. Line quality has been presented in both qual-

itative and quantitative forms. Finally, the manner in which an illustrated report was obtained on the 3800 has been described in both text and graphics.

The experimental system employs a batch mode program driven by a list language. Further study is needed on an interactive program for the display of character-graphic images at a terminal. Experiments on graphics definition and editing of the input (rather than list-language definition) could lead to an additional enhancement.

If more characters were devoted to graphics it would be possible to reproduce lines at other slopes exactly (smoothly), rather than in a minimum-error approximation. Doubling the number of graphic characters would permit a second line weight to be supported. The centering of text within flow chart boxes could be achieved by more powerful software. The goal of this experiment has been to show capability, not ultimate performance. The balance selected between text and graphic character count permits fast, economical reproduction of illustrated reports.

CITED REFERENCES

1. *Business Graphing on the IBM 3800.* IBM Systems Library, order number SE21-2168-0 (1977), available through IBM branch offices.
2. *IBM 3800 Plotting Facility*, IBM Systems Library, order number SB21-2155-0 (1977), available through IBM branch offices.
3. *Reference Manual for the IBM 3800 Printing Subsystem*, IBM Systems Library, order number GA26-1635-2 (July 1978), available through IBM branch offices.
4. J. E. Bresenham, *A Linear Incremental Algorithm for Digitally Plotting Circles*, Technical Report TR02.286, IBM General Products Division, San Jose, CA 95193 (January 27, 1964).
5. D. Cohen, "On linear difference curves," *Proceedings of the International Symposium CG70* **1,** Brunel University, Uxbridge, England (April 1970).
6. D. Cohen, *Incremental Methods for Computer Graphics*, Technical Report ESD-TR69-193, Harvard University, Cambridge, MA (April 1969).
7. P. E. Danielsen, "Incremental curve generation," *IEEE Transactions on Computers* **C-19,** No. 9, 783-793 (September 1970).
8. E. A. Denert, "A method for computing points on a circle using only integers," *Computer Graphics and Image Processing* **2,** No. 1, 83-91 (August 1973).
9. B. W. Jordan, W. J. Lennon, and B. C. Holm, "An improved algorithm for the generation of nonparametric curves," *IEEE Transactions on Computers* **C-22,** No. 12, 1052-1060 (December 1973).
10. R. A. Metzger, "Computer generated graphic segments in a raster display," *AFIPS Proceedings of the Spring Joint Computer Conference, 1969*, AFIPS Press, Montvale, NJ (1969), pp. 161-172.
11. M. L. V. Pitteway, "Algorithm for drawing ellipses or hyperbolae with a digital plotter," *Computer Journal* **10,** No. 3, 282-289 (November 1967).
12. M. L. V. Pitteway, "Integer circles, etc.—three more extensions of Bresenham's algorithm," *Computer Graphics and Image Processing* **3,** No. 3, 260-261 (September 1974).
13. M. L. V. Pitteway, "Integer circles—some further thoughts," *Computer Graphics and Image Processing* **3,** No. 3, 262-265 (September 1974).

14. Y. Suenaga, T. Kamae, and T. Kobayashi, "A high-speed algorithm for the generation of straight lines and circular arcs," *IEEE Transactions on Computers* **C-28,** No. 10, 728–736 (October 1979).
15. H. Freeman, "Algorithm for generating a digital straight line on a triangular grid," *IEEE Transactions on Computers* **C-28,** No. 2, 150–152 (February 1979).
16. J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal* **4,** No. 1, 25–30 (1965).
17. J. E. Bresenham, "A linear algorithm for incremental digital display of circular arcs," *Communications of the ACM* **20,** No. 2, 100–106 (February 1977).

The author is located at the IBM General Products Division Santa Teresa Laboratory, 555 Bailey Avenue, San Jose, CA 95150.

Reprint Order No. G321-5129.