Comput. & Educ. Vol. 5. pp. 193 to 200. 1981
Printed in Great Britain

0360-1315/81/040193-08$02.00/0
Pergamon Press Ltd

LOW COST COMPUTER GRAPHICS IN ENGINEERING EDUCATION

RICHARD L. PHILLIPS

Computer, Information, & Control Engineering, The University of Michigan, Ann Arbor, MI 48109, U.S.A.

Abstract—The Apple II is a personal computer which provides 6 color, raster graphics via a 280 h × 192 v frame buffer. A user-programmable MOS Technology 6502 CPU addresses 48 K bytes of RAM, 12 K of ROM, and 4 K of memory-mapped I/O. This system has been evaluated in a variety of applications for which one normally assumes higher resolution, higher cost graphics devices are needed.

Examples were drawn from computer aided design, color microfilm previewing, cartography, and aircraft synthesis. The results were encouraging; low resolution raster scan graphics is surprisingly effective for a wide range of applications. As a result, a computer graphics laboratory, with a cluster of 10 Apples as its nucleus, was formed and has been functioning successfully for 4 terms.

INTRODUCTION

Overview

There are never enough terminals—a familiar refrain for anyone who teaches interactive computer programming. The terminal shortage is especially acute for courses in computer graphics, mostly because each unit costs at least $5000. I teach a course in "highly interactive" computer graphics where a student will spend between 30–40 h at a terminal for each problem. With a class size of 50 students, the demand for terminal access is feverish.

Several months ago, I set out to remedy the terminal shortage problem by investigating the personal computer as a possible display device. The main attraction of personal computers is their price. If they can be made to do useful work, there is a real potential for cost effectiveness. In our context, usefulness is measured in terms of demanding scientific problem solving for teaching and research. Computer graphics plays an important role in this process. This paper presents the results of a several month evaluation of the potential of a personal computer for doing serious work.

The Apple II computer

One has only to leaf through a recent copy of BYTE magazine to appreciate the bewildering array of personal computers now available. The "big three" among these are the TRS-80, Commodore Pet and the Apple II. The Apple II was selected for our purposes because it assumes a minimum user naivete and provides true computer graphics capability (beware of alphanumeric manipulations masquerading as graphics). Other Apple II features that appeal are:

1. a versatile resident monitor.
2. 48 K bytes of user-programmable RAM.
3. 12 K bytes of user-redefinable ROM.
4. two 8 K byte frame buffers for defining raster scan images of 280 h × 192 v resolution. These frame buffers can be context switched under program control to produce animation effects.
5. a full BASIC interpreter with graphics extensions.
6. a variety of peripherals including a floppy disk, printer, light pen, PROM programmer, and communications interface.

The basic Apple II price is about $2000 including a communications interface and a floppy disk. While this is not exactly an "out of pocket" amount, it is nonetheless an attractive price to an educator who would otherwise have to pay about $5000 (Tektronix 4010) simply for a graphics terminal with no computing power.

Project goals

The goals of the evaluation project were to:

1. Determine the range of applications for which the low, 280 × 192 graphics resolution is acceptable.

2. Evaluate the stand alone computing power of the Apple II for "real" problems.
3. Study the Apple II as a distributed computing element in a time sharing system.

STAND-ALONE CAPABILITIES

Capacity and speed

The full BASIC interpreter has a complete complement of string capabilities, mathematical functions, and graphics commands. In principle, any problem that is solved on a large system can be solved on the Apple. There are, however, the practical limitations of program size and speed of solution to consider. Even with 48 K bytes of user RAM, indefinitely large programs cannot be constructed. But Apple BASIC uses dynamic memory management (better than FORTRAN!) and stores the program text in a compressed, tokenized form so that 100 variable, 1000 statement programs are certainly feasible. Speed is another thing, however. As with any interpreter, speed of execution can be somewhat slow, requiring perhaps minutes for a problem that would take a few seconds on a large system. (This comparison may not be too unfavorable, however, if one is dealing with a heavily loaded host.)

Examples

Two problems from computer aided design were chosen to exemplify stand-alone capabilities of the Apple II. These problems, as others described later, have a heavy graphics emphasis, both on input and output. Both examples are described by Chasen[1] and are representative components of more extensive programs used in industry.

Often one wishes to approximate roughly defined two-dimensional contours by a series of blended analytic functions; ellipses are a common choice. Figure 1* shows the layout of such a problem. The user has been prompted to specify Y1 and X3, the points where the two ellipses will cross the y and x axes, respectively. The line y = Y2 is the ordinate of the blending point and X2B is the minimum abscissa where the blending may occur. Once the x value is selected, blending can take place within a range of limiting slopes. At this point the two ellipse segments are drawn.

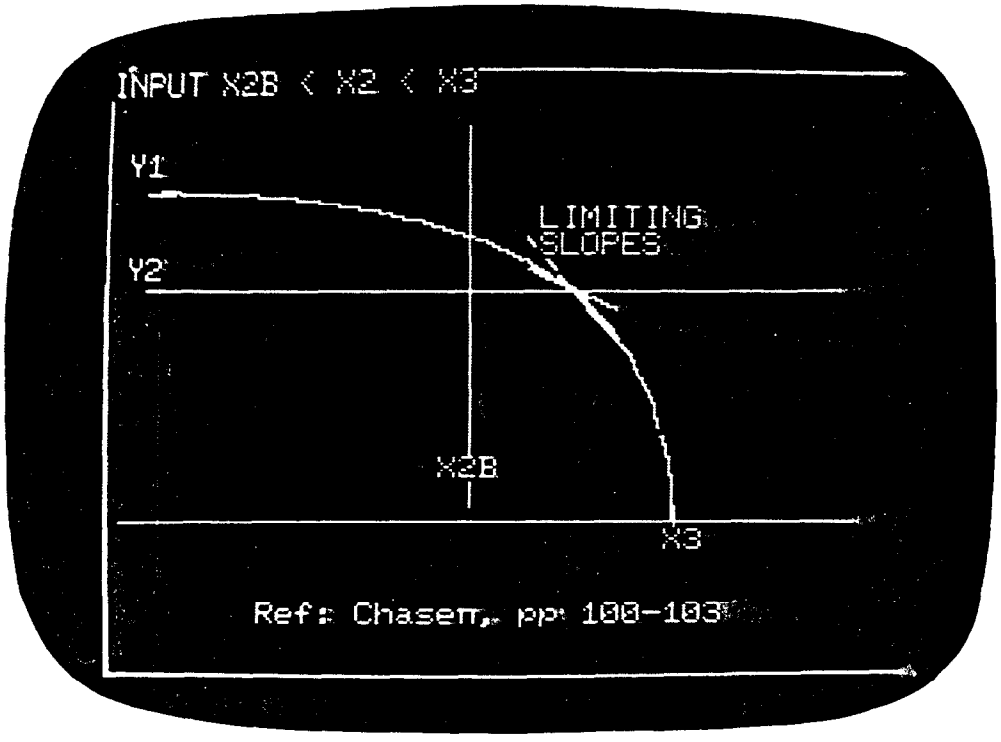

Fig. 1. Blended ellipses.

* All figures have been produced by directly photographing the Apple display screen.

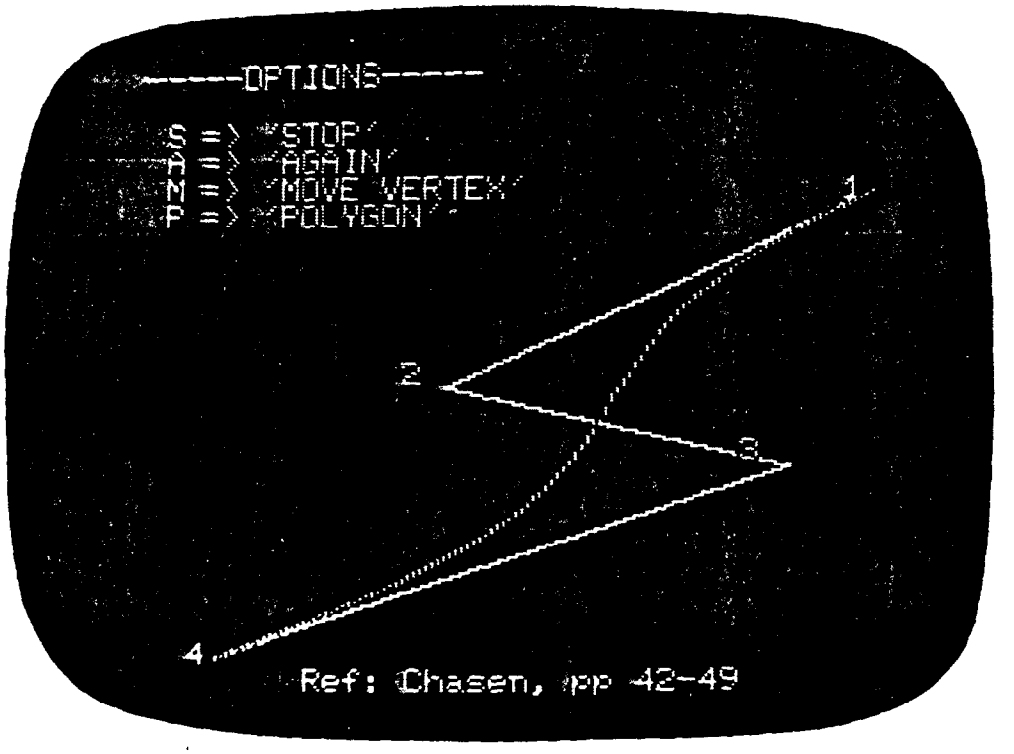

Fig. 2. Bezier polynomial.

All input necessary to produce Fig. 1 was done graphically by rapidly moving a non-destructive cursor to the desired point and striking a key. The curves are computed and displayed in less than a second, making the entire process highly interactive. On a color display the two ellipses are drawn in red and blue to facilitate distinction.

Another example of two dimensional curve manipulation is shown in Fig. 2. Bezier polynomials are defined by a control polygon; points 1–4 denote its vertices. The Bezier polynomial thus determined will be a smooth curve passing through the first and last vertices where it is also tangent to the initial and final line segments. By moving any of the control vertices, the designer can rapidly develop the desired curve; the value of the technique depends on a high degree of interactivity.

In Fig. 2 the control vertices were imput graphically. The computation of the Bezier curve requires repeated evaluation of the factorial function and for the case shown required about 10 s to compute and draw. This is too slow to be considered interactive. One would like to be able to move a selected vertex around and have the new Bezier curve follow it almost immediately. Here is a case, then, where the computing power of the Apple II (really the BASIC interpreter) is not sufficient to the task. The situation could be improved if a compiled program were being run or, of course, if the time critical portions of the algorithm were coded in 6502 assembly language.

Yet another possibility is to program the Apple to behave as a graphics terminal, do all computation on a large host system, and send the coordinates of the computed curve to the Apple. Unless the communications bandwidth is sufficiently high, however, plotting the curve can be as slow as doing the entire task on the Apple. An alternative possibility is to devise a division of labor between the host and the Apple that optimizes interactivity. This approach is discussed later.

TERMINAL CAPABILITIES

Communicating with a host

With little difficulty the Apple II can be programmed to function as a computer terminal, provided a communications interface is available. There are several suppliers for these boards; they range in price from $60 to $180. Because of the farsighted Apple design it is possible to access the communications board from a BASIC program as well as directly from an assembly language program. Thus, all sorts of information transfer possibilities exist; sending data and programs back and forth between host and terminal; running an Apple program that reads data from a host file, and vice-versa; and

down loading a cross-assembled program from host to Apple. All of the above examples suggest transfer of just character information. Since the Apple II has graphics capabilities, however, its use as a graphics terminal has been investigated.

Simulating a Tektronix terminal

Apple II computer graphics is of the raster scan[2] variety. An 8 K byte region of RAM is designated as the graphics bit map or frame buffer. The frame buffer can be thought of as a memory mapping of the video screen. Each point on the screen that is illuminated corresponds to a bit being "on" in the frame buffer. The resolution afforded by an 8 K frame buffer corresponds to approx. 280 columns × 192 rows of potentially lighted dots, or pixels, on the screen. By context switching, the Apple can be directed to graphics mode whereby its video output reflects the contents of the frame buffer. Since the frame buffer is user RAM it can be written in by a program. In fact, a line between any two points in the 280 × 192 grid is produced by turning on those pixels in the frame buffer that give the best approximation to the line. The user need not do this directly; low level software is provided for this purpose. To operate as a graphics terminal, then, data sent by a host computer that describes a line must be passed to the Apple software for pixel mapping in the frame buffer.

The line-producing data sent by the host could be designed specifically for an Apple, but this would be restrictive and short-sighted. The Apple could function as a graphics terminal only on those host computers having that special program. Rather, the approach was to develop a program for the Apple that causes it to emulate all terminals in the Tektronix 4010 family. Software to drive Tektronix terminals abounds; any host computer that supports graphics probably supports the Tektronix. It matters not whether the resident software is PLOT-10 (a Tektronix product) but only that the data stream sent to the terminal is in Tektronix format[3].

A ROM-based interpreter called TEKSIM* allows the Apple II to behave as a Tektronix terminal. TEKSIM is sensitive to the special mode characters that cause a Tektronix to operate in alphanumeric mode, graphics mode, graphics input mode, etc. Naturally, not all Tektronix features can be emulated, most notably the resolution which ranges from 1024 × 1024 to 4096 × 4096. On the other hand, the Apple has features such as color, which amount to an enhancement of graphics capabilities. These will be discussed later.

Figures 3 and 4 are examples of Apple emulations of graphics output that was originally designed for Tektronix terminals. The range of applications for which Apple graphics resolution is acceptable is clearly depicted. Figure 3 is typical output from a general aviation synthesis program [4] and shows a three-view of a proposed airplane design. Figure 4 is an example from cartography and shows that distinguishing features of counties and lakes are clearly identifiable.

Extensions to Tektronix software

The Apple II can produce graphics in 6 colors and TEKSIM supports this capability. To use this feature requires modification of host computer software because existing Tektronix programs would not normally provide for color selection. This modification is usually a trivial task, however, because to send color information to the Apple through TEKSIM requires only that 3 characters be sent by the host. In line with the Tektronix convention, these are an ESC (escape), DC2 (device command 2), and an ASCII integer from 0–5. Any lines drawn after this command will be in the color specified by the third character (a look-up table maps this integer into a corresponding color). Substituting a DC3 for the second character in the above sequence causes the screen background to be set to the specified color upon next erasure. Tektronix PLOT-10 software provides an easy way to send such character sequences via the TOUTST subroutine [5]. Other host graphics libraries provide similar capabilities.

Figure 5 is an example of using the color select features (reproduced here in black and white) of TEKSIM where the Apple was used to preview a slide that was to be produced on an FR-80 film recorder. The pie chart in Fig. 5 was shaded by individual lines and required many minutes of connect time at 300 baud. Since a sector of a circle can be specified by center location, radius, and limiting angles, only that information need be sent to the Apple and the shading could be done locally. This leads to the notion of distributed computing.

DISTRIBUTION OF TASKS

Graphic output primitives

In deciding on tasks to distribute among computing elements, one criterion is to select those operations that do not require feedback. That is, a small amount of information can be sent from host

* TEKSIM is available from ABW Corporation, Ann Arbor, MI. EDUCOM members are entitled to a special purchase arrangement.

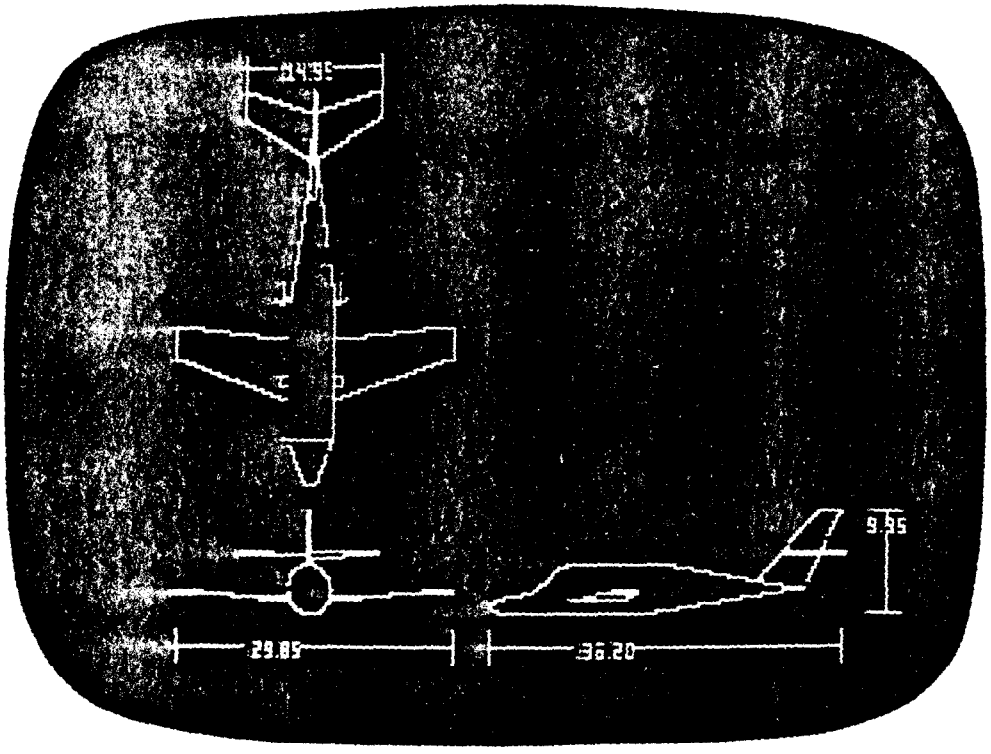

Fig. 3. Airplane synthesis

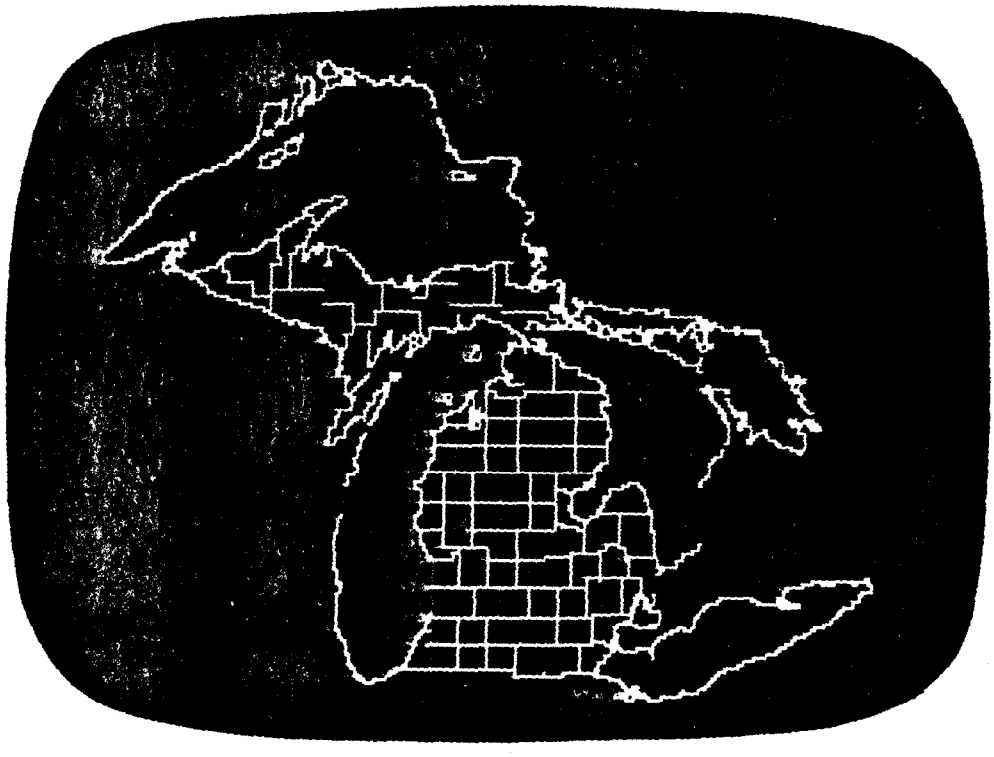

Fig. 4. Cartographic example.

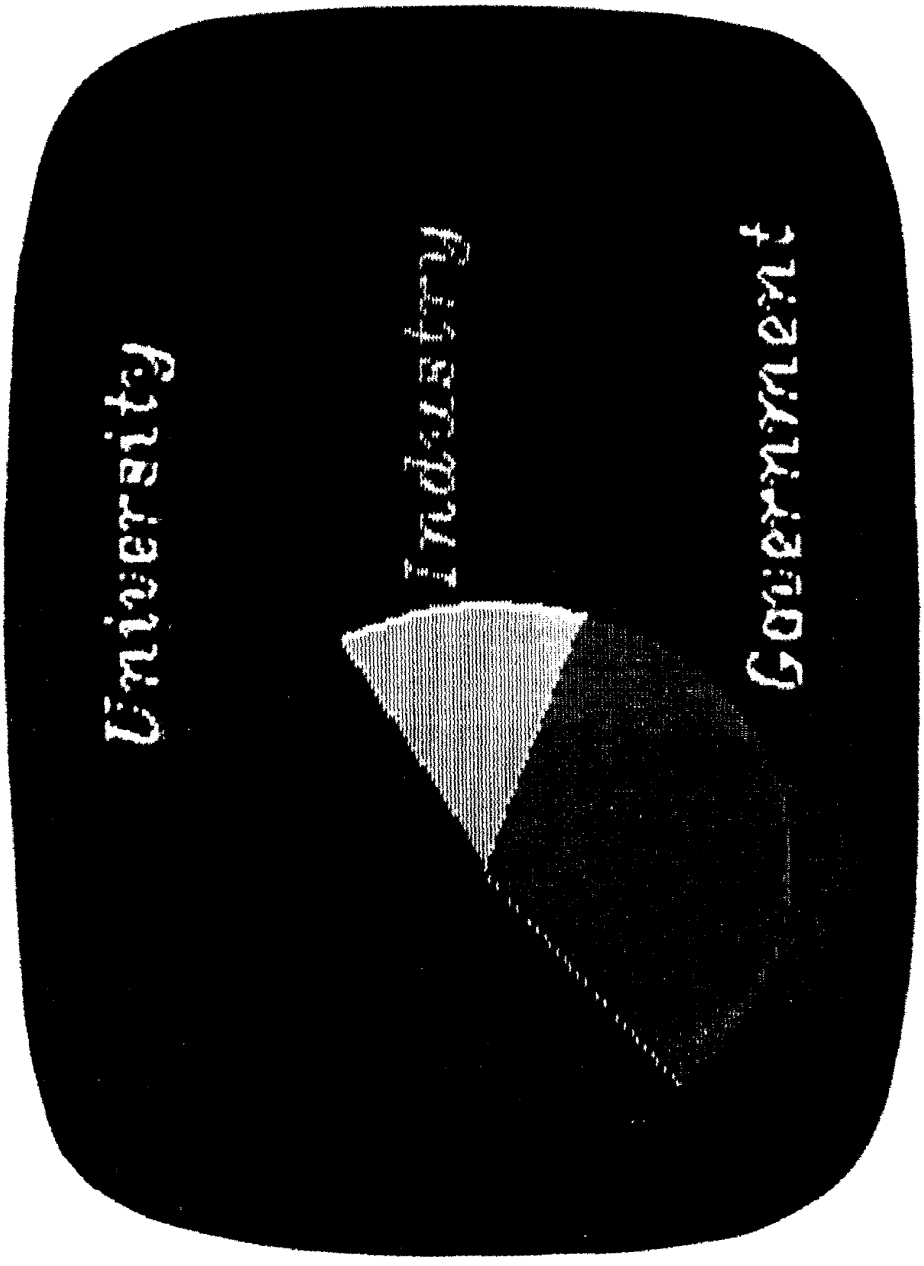

Fig. 5. Color slide preview.

to terminal, say, and the task specified can go to completion without further communication. In graphics there are many candidates for such tasks; they involve inherently graphical operations and we call them graphic primitives. Obvious candidates for primitives are a rectangle, a circle, conic sections, etc. Shaded figures such as bars, pie charts, and general polygons are even more useful primitives because of the time involved in the shading process. Several of these primitives have been implemented in conjunction with TEKSIM resulting in a significant speed increase over straight terminal performance. So far the bar, circle, and pie chart have been implemented as primitives. This means that when a certain character sequence is received from the host, TEKSIM jumps to an external routine to produce the requested graphical output. An example of such a routine is the one that implements the algorithm of Badler [6] for drawing a shaded circle. This algorithm requires only addition, subtraction and shifting to produce a pie chart, so it is ideal for a microprocessor environment. The algorithm is fast; for the Apple a disk covering the entire screen is drawn in less than a second. By contrast, sending the shading lines from a host, using the same algorithm, requires 5 min!

Additional output primitives are planned for use with TEKSIM. In addition to a shaded general polygon, an axis generation routine for graph drawing is being developed. By sending only the extrema of the dataset to be plotted, the primitive will lay out an axis system, determine "nice" scale intervals for each axis, and apply corresponding tick marks and labelling. Then, one or more datasets can be plotted using different symbols or line colors.

Graphic input primitives

TEKSIM supports graphic input through the use of a joystick which propels a non-destructive tracking cross around the screen. Through a software "hook", however, any other graphic input device can replace the joystick. Upon command from the user an appropriate machine language module is downloaded from the host. TEKSIM then accesses this module whenever the host application program requires graphical input. In this way we can support graphic input from a light pen, a variety of graphic tablets and, for a handicapped student who can only use a mouth stick, from the keyboard.

CONCLUDING REMARKS

Findings

There is no doubt that the Apple II can do serious, constructive work in an educational and research environment. The notion of task distribution has proven extremely useful, especially for computer graphics. Moreover, the graphics resolution of the Apple II appears to be satisfactory for a wide range of applications. Only a few cases have been encountered where information is lost because of resolution.

The Apple II graphics Laboratory

Armed with the above findings and a modest amount of money we marshalled all the resources needed to assemble a computer graphics laboratory with a nucleus of ten Apple II computers. All Apples are equipped with TEKSIM and are connected to our Amdahl 470V/8 through a special front-end computer called a *remote data concentrator* (RDC). Each Apple communicates with the host at 9600 baud and has its own floppy disk for local file storage. The laboratory also includes a CORVUS 11A 10 mb hard disk system. All Apples in the cluster can communicate with a special "disk mother" Apple through the RDC for rapid file transfer. The Corvus system provides the equivalent of 87 Apple disks of on-line storage. Finally, a specially modified Houston Instrument COMPLOT electrostatic copier provides the cluster with hard copy capability. Each Apple has direct queued access to the copier through remote switching.

The future

This is really just a beginning. While the laboratory was initially established to teach computer graphics, it has been used by many courses having both graphical and non-graphical requirements. It is widely used by freshman computing classes, by design classes and for teaching engineering graphics.

Being interested in computer graphics, I am looking forward to enhancing the capabilities of the cluster beyond just emulating Tektronix terminals. This involves using the local processing power of the Apple to do such things as selective erase, dynamic graphics, local zoom and pan, graphics scrolling, and improved color manipulation.

I expect to see this cluster concept replicated manyfold, not just at Michigan but elsewhere. It represents considerable capability for a modest investment.

REFERENCES

1. Chasen S. H., *Geometric Principles and Procedures for Computer Graphics Applications.* Prentice–Hall, Englewood Cliffs, NJ (1978).
2. Newman W. M. and Sproull R. F., *Principles of Interactive Computer Graphics,* Second Edition, Chapters 15–19. McGraw–Hill, New York (1979).
3. Anon, *Tektronix 4014 and 4014-1 Computer Display Terminal Users Instruction Manual.* Tektronix Inc., P.O. Box 500, Beaverton, OR.
4. Galloway T. L. and Smith M. R., General aviation design synthesis utilizing interactive computer graphics. Society of Automotive Engineers Business Aircraft Meeting, Wichita, KS (1976).
5. Anon, *Tektronix PLOT-10 Terminal Control System Users Manual.* Tektronix Inc., P.O. Box 500, Beaverton, OR.
6. Badler N. I., Disk generators for a raster display device. *Comput. Graphics Image Process.* **6**, 589 (1977).