Reflections Of A Game Designer: Author Michael Crichton

COMPUTE!

$2.95
February
1985
Issue 57
Vol. 7, No. 2
$3.50 Canada
02193
ISSN 0194-347X

The Leading Magazine Of Home, Educational, And Recreational Computing

Special Games Issue

Ready-To-Use Software
For Commodore 64, VIC-20, Atari, Apple, IBM

Advanced Sound Effects On The Commodore 64

Plus/Term Modem Program For 64 & VIC-20

Adding Sound Effects To Atari

Acrobat: Exciting Animated Game For 64, VIC, Atari

The New Atari: COMPUTE! Interviews Sigmund Hartmann

71486 02193

February 1985 Vol. 7, No. 2

NOTE: See page 145 before typing in programs.

AP Apple, **Mac** Macintosh, **AT** Atari, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **CC** Radio Shack Color Computer.
*All or several of the above.

TOLL FREE Subscription Order Line
800-334-0868 (In NC 919-275-9809)

COMPUTE! Publications,Inc.
One of the ABC Publishing Companies

One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. Send subscription orders or change of address (P.O. form 3579) to **COMPUTE!** Magazine, P.O. Box 914, Farmingdale, NY 11737. Second class postage paid at Greensboro, NC 27403 and additional mailing offices. Entire contents copyright © 1984 by COMPUTE! Publications, Inc. All rights reserved, ISSN 0194 357X.

READERS' FEEDBACK

The Editors and Readers of COMPUTE!

Do You Need Two Disk Drives?

I have a TI-99/4A and an Amdek color monitor. I'm planning to buy a second computer, but I'm confused about the advertising for two disk drives. Why do you need two disk drives?

Natalia Macedo

For many home computer users, one disk drive is sufficient. Two drives are, however, useful in several ways. For example, if you do a lot of disk copying you won't have to keep swapping the disks back and forth.

Some software packages, particularly business programs, require two disk drives. An example would be a data base manager which holds the program disk in one drive and the data disk in the other. Similarly, if you do a lot of programming, you can use two drives to hold your system disk and utilities. Some compilers and assemblers also require or work much faster with two drives.

Many writers prefer dual-disk systems for word processing because they can periodically save backups on separate disks for safety. In general you can live with one and live very well with two.

What Is DOS?

I'm going to buy a disk drive, and different brands of drives have a different DOS. What exactly is DOS?

Ricky Gibbs

DOS (usually pronounced to rhyme with "moss") stands for Disk Operating System. *Basically, this is a program which allows the computer to work with a disk drive. On most computers, DOS lets you save and load files, view disk directories (lists of files stored on disks), rename files, erase files, copy files from one disk to another, copy entire disks, format blank disks (prepare them for use), and other functions.*

There are many different types of DOS for different computers, and they're usually incompatible with each other. It's important that you use the proper DOS for your computer, disk drive, and system configuration. Fortunately, most disk drives (or computers with built-in disk drives) already include the proper DOS.

Usually DOS comes on a disk that must be inserted in the disk drive before you turn on the computer. It loads automatically when the power is switched on. This process is called booting up. *An exception is Commodore DOS, which is stored in Read Only Memory (ROM) chips within the disk drive itself. Commodore DOS is available whenever the computer and disk drive are powered up.*

There are many versions of DOS even for the same computer. As revisions, corrections, and updates are made, new versions of DOS are released, usually denoted by different numbers. Examples are Atari DOS 1 (the original version), DOS 2.0S (improved single-density), and DOS 3 (enhanced density); PC-DOS 1.1 (the original version), DOS 2.0 (with improvements added for hard disks), and DOS 2.1 (modified for the PCjr); Apple DOS 3.3 (originally intended for the Apple II and II+) and ProDOS (introduced with the Apple IIe and IIc); and so on. Commodore DOS is harder to modify since it's embedded in ROM chips, but unofficial updates are usually made when new models of disk drives are introduced.

In addition to the DOS versions released by computer manufacturers, there are also custom versions of DOS sold by independent companies for certain computers. Examples are OS/A+ DOS for Atari computers, CP/M-86 for IBM computers, and CP/M-80 for numerous personal computers. Sometimes a custom DOS is compatible with the manufacturer's DOS, and sometimes it requires extra hardware (such as a CP/M board).

The disk drive you buy for your computer will probably come with the right DOS for your system. If it doesn't, the dealer can recommend the proper DOS or a compatible custom DOS.

The Great Commodore Save/Replace Debate

I have a Commodore 64 and a 1541 disk drive. Recently I saved a program on a disk and later saved another program on the same disk. When I

Thanks for sharing the information. It's also helpful to begin your program with a loader routine that installs DOS 5.1 so you don't have to remember to load the Wedge each time. In addition, this keeps other people who may use your program from complaining about the inexplicable crashes they may encounter otherwise. Make sure DOS 5.1 is on the same disk as your loader routine.

TI POKE?

I own a TI-99/4A and would like to know if there is an equivalent for POKE in TI BASIC or TI Extended BASIC.

Paul Parks

There is no equivalent for POKE or PEEK in standard TI BASIC. This is one example of how TI BASIC differs considerably from other personal computer BASICs. The language designers may have felt that PEEK and POKE commands—which allow programmers to examine and modify individual memory locations—were somehow risky tools to put in the hands of inexperienced programmers. Of course, many inexperienced programmers progress beyond that stage and would find uses for these commands. Constructing a blockade between the programmer and the lower levels of the machine can severely limit a user's control.

Fortunately, TI's Extended BASIC does provide an equivalent for POKE, the CALL LOAD statement. For example, to place the value 100 in location 20000, you'd use CALL LOAD(20000,100). The equivalent to PEEK is CALL PEEK. To place the value from location 20000 into the variable X, you'd use CALL PEEK(200000,X).

Remember that memory for the video display is maintained separately from the microprocessor (and, without expansion, BASIC programs are actually stored in the video memory area), so CALL LOAD and CALL PEEK give you access only to the processor memory or to any attached expansion memory. The ROM in the Mini Memory *cartridge also provides for CALL LOAD and CALL PEEK, and in addition provides CALL POKEV and CALL PEEKV, which allow you to store and retrieve data from video memory.*

TI also supplies an impressive library of built-in subroutines that accomplish many of the things that PEEK and POKE are used for on other computers. For example, to read the TI joysticks, you can type:

```
100 CALL JOYST(1,X,Y)
```

Other valuable features are CALL CHAR, RESEQUENCE, and NUMBER. These provide built-in character redefinition, renumbering, and automatic line-numbering utilities.

Apple Shape Tables

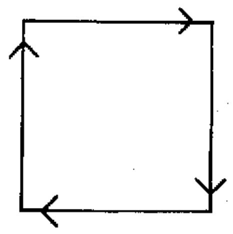
I have an Apple II+ and have been trying to figure out shape tables. How do the data numbers affect a shape? How do the numbers in the DRAW and XDRAW commands make a shape?

Tony Steele

Shape tables can appear very confusing, but they are extremely useful, though in some cases it may be easier to draw complicated figures with HPLOT.

Basically, a shape table contains plotting vectors to draw a figure. Each vector describes the movement necessary to draw the object.

Let's try constructing a shape table to draw a square to see how it all gets done. The first step is to draw the shape on a piece of paper.



Now you must convert the figure to coded plotting vectors. Vector codes are numbers between 0 and 7 which correspond to a direction of movement, and each byte of a shape definition can hold as many as three vectors. The task now is to reduce the shape to a series of vectors, then place these vectors into memory, where they can be used to draw shapes.

Pick a starting point on the figure you want to code. For our square, we'll start at the bottom-left corner. Make a list of the directions required to draw the shape. Be sure you include all movements necessary, even those not actually drawn on the screen.

Starting at the bottom-left corner, we need these vectors to draw our square:

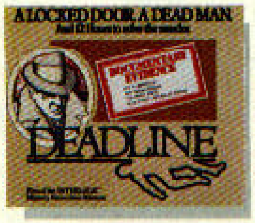
Vector	Plot
up	yes
right	yes
down	yes
left	yes

Now use this table to write the proper binary code next to each vector:

Action	Binary Code	Decimal Code
move up without plotting	000	0
move right without plotting	001	1
move down without plotting	010	2
move left without plotting	011	3
move up with plotting	100	4
move right with plotting	101	5
move down with plotting	110	6
move left with plotting	111	7

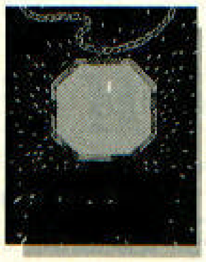
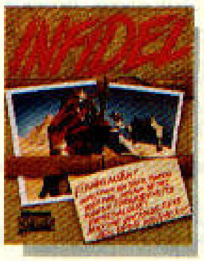
ZORK I
INFOCOM

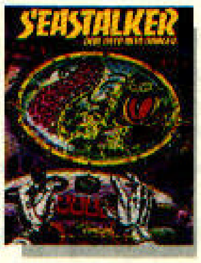
ZORK II
INFOCOM

ZORK III
INFOCOM

A LOCKED DOOR. A DEAD MAN.
DEADLINE

WITNESS
SOMEBODY'S GOING TO TAKE THE DEEP SIX!

STARCROSS
INFOCOM



AND NOW FOR SOMETHING INCOMPLETELY DIFFERENT!

Incomplete, yes.
But it's not just because we're always bringing out new stories in the Infocom interactive fiction collection. Nor is it simply due to the fact that with all the writing and re-writing, honing and perfecting that we put into every one of our stories, our work is seemingly never done.

The real reason is: an Infocom work of fiction can never be complete until you become a part of it.

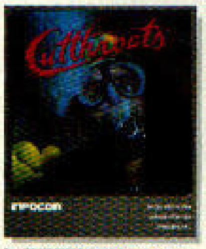
You see, as hard as we work at perfecting our stories, we always leave out one essential element—the main character. And that's where you enter in.

Once you've got Infocom's interactive fiction in your computer, you experience something akin to waking up inside a novel. You find yourself at the center of an exciting plot that continually challenges you with surprising twists, unique characters (many of whom possess extraordinarily developed personalities), and original, logical, often hilarious puzzles. Communication is carried on in the same way as it is in a novel—in prose. And interaction is easy—you type in full English sentences.

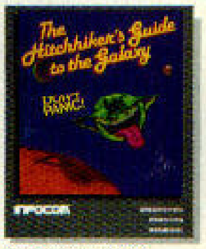
But there is this key difference between our tales and conventional novels: Infocom's interactive fiction is active, not passive. The course of events is shaped by the actions you choose to take. And you enjoy enormous freedom in your choice of actions—you have hundreds, even thousands of alternatives at every step. In fact, an Infocom interactive story is roughly the length of a short novel in content, but because you're actively engaged in the plot, your adventure can last for weeks and months.

In other words, only you can complete the works of Infocom, Inc. Because they're stories that grow out of your imagination.

Find out what it's like to get inside a story. Get one from Infocom. Because with Infocom's interactive fiction, there's room for you on every disk.



In CUTTHROATS,™ the plot involves a motley band of hardbitten salts who get wind of a shipwreck laden with sunken treasure near the remote island where you live. In exchange for your diving skills, they offer you a piece of the action. Your challenge: survive them, the perils of the deep, and escape with the treasure and your life. Good luck!



THE HITCHHIKER'S GUIDE TO THE GALAXY™ by Douglas Adams is the most mind-boggling story we've ever published. In the person of Arthur Dent, you'll chortle as your planet is demolished. You'll yelp with laughter as your life is threatened by a galaxy of horrors. Your sides will positively split as you search the universe for . . . well, you'll find out. Maybe.

In SUSPECT,™ our newest mystery thriller, you're a reporter who gets the scoop on the society event of the year—the murder of a Maryland Blue Blood at a fancy costume ball. Great! Except you're the prime suspect. And if you can't find the real killer, your next by-line could be in the obituaries.

INFOCOM™

Infocom, Inc., 55 Wheeler Street, Cambridge, MA 02138

For your: Apple II, Atari, Commodore 64, CP/M8", DECmate, DEC Rainbow, DEC RT-11, IBM PC* and PCjr, KAYPRO II, MS-DOS 2.0* NEC APC, NEC PC-8000, Osborne, Tandy 2000, TI Professional, TI 99/4A, TRS-80 Models I and III.
*Use the IBM PC version for your Compaq, and the MS-DOS 2.0 version for your Wang or Mindset.

Apple Emulator For Commodore 64?

I have heard of an add-on system for the Commodore 64 that will allow you to use any of the Apple hardware and software on the 64. What is it?

Jason Meudt

When the 64 was first introduced, there wasn't very much software available for it, but Apple had thousands of programs available on just about every subject imaginable. It wasn't long before rumors began circulating about an Apple emulator which plugged into the 64 and turned it into an instant Apple. Some companies even advertised them and took orders. As far as we know, none was actually delivered.

The problem of one computer emulating another is complex. Besides having to duplicate the functions of the operating system of the computer being emulated, you must also have a disk drive which can read the other system's disks. Commodore's 1541 normally can read only those disks formatted on disk drives compatible with the 1541, not Apple disks. Hence, you'd need an Apple-compatible disk drive. Even though both the Apple and Commodore use a 6502-family microprocessor, you must still have Apple DOS and a different operating system. All that remains of your original 64 is the keyboard, some RAM chips, and the microprocessor. Therefore, an Apple emulator for the 64 would end up costing almost as much as an Apple purchased outright.

There's also a possible legal complication. Apple has been very aggressive in bringing lawsuits against vendors who market products with ROMs that Apple feels are close copies of its own operating system. For example, Apple successfully fought a long legal battle with the makers of the Franklin Ace computer. Since the emulator would have to provide an operating system that closely resembled Apple's, it's quite possible that the manufacturer would end up in court.

Moreover, new programs for the 64 have been published or released commercially on almost a daily basis since the 64 was introduced. By now most of the original Apple library has been translated for the 64, with enhancements to take advantage of the 64's more advanced sound and graphics capabilities. Thus, much of the original impetus for the development of an emulator has dwindled. In fact, with the booming library of original software for the 64, a 64 emulator for the Apple might prove more popular.

Nevertheless, one Apple emulator is currently being advertised in COMPUTE!, *though at this writing it is not yet available. Mimic Systems Inc., 1112 Fort St., Fl. 6M, Victoria, B.C., Canada V8V 4V2, has announced an Apple emulator and plans to have it ready for the Winter Consumer Electronics Show in January, with sales to begin early in 1985. Mimic's current price estimate is around $600. For comparison, the Apple IIe is presently available for about $800.*

Program Your Own EPROMS

▶ VIC 20™
▶ C 64™

$99.50

PLUGS INTO USER PORT.
NOTHING ELSE NEEDED.
EASY TO USE. VERSATILE.

promenade™

- Read or Program. One byte or 32K bytes!
- OR Use like a disk drive. LOAD, SAVE, GET, INPUT, PRINT, CMD, OPEN, CLOSE—**EPROM FILES!** Our software lets you use familiar BASIC commands to create, modify, scratch files on readily available EPROM chips. Adds a new dimension to your computing capability. Works with most ML Monitors too.
- Make Auto-Start Cartridges of your programs.
- The *promenade*™ C1 gives you 4 programming voltages, 2 EPROM supply voltages, 3 intelligent programming algorithms, 15 bit chip addressing, 3 LED's and NO switches. Your computer controls everything from software!
- Textool socket. Anti-static aluminum housing.
- EPROMS, cartridge PC boards, etc. at extra charge.
- Some EPROM types you can use with the *promenade*™

2758	2532	462732P	27128	5133	X2816A*
2516	2732	2564	27256	5143	52813*
2716	27C32	2764	68764	2815*	48016P*
27C16	2732A	27C64	68766	2816*	

▶ ™Commodore Business Machines

*Denotes electrically erasabl types

Call Toll Free: 800-421-7731
In California: 800-421-7748

JASON-RANHEIM
580 Parrott St., San Jose, CA 95112

VISA

TI Rounding Routine

Here's a routine for TI-99/4A users that will round off decimal points to any desired place.

Bill Gardella

```
100 N=0
110 D$=STR$(C)
120 N=N+1
130 E$=SEG$(D$,N,1)
140 IF E$="" THEN 250
150 IF E$<>"." THEN 120
160 E$=SEG$(D$,1,N+2)
170 F$=SEG$(D$,N+3,1)
180 IF F$="" THEN 200
190 G=VAL(F$)
200 H=VAL(E$)
210 IF G<5 THEN 230
220 H=H+.01
230 C=H
240 GOTO 270
250 H=VAL(D$)
260 C=H
270 REM Rest of program from here o
    n.
```

Thanks for sharing the technique.

THE BEGINNER'S PAGE

Tom R. Halfhill, Editor

IF-THEN Intelligence

At one time or another you've probably seen the term *artificial intelligence*. It refers, of course, to computer intelligence—the ability of a machine to reproduce (or, if you prefer, simulate) some of the thought processes of a human being.

We're not going to reopen here the philosophical debate about whether computers are really intelligent, or if they ever will be intelligent (or for that matter, if humans are intelligent). Scientists still can't agree on exactly how the human brain works, much less whether it can be duplicated in silicon circuitry.

But we do know how computers work. Although computers aren't (yet) capable of independent thought or action, they certainly appear intelligent at times. They can play chess at the grandmaster level, forecast tomorrow's weather as well as anybody else, help plan the economy of a household or a nation, create wonderfully abstract art, and even simulate the responses of a psychoanalyst closely enough to fool many laypeople. How can a mass of wires and silicon chips seem to be so smart? What sets computers apart from all other machines?

Programmability alone isn't the answer. There were programmable machines long before computers came along. One example is the centuries-old music box. A melody is programmed into the box by punching little bumps onto the surface of a revolving drum; as the drum turns, the bumps pluck a series of tiny metal prongs tuned to different notes. Of course, a music box is capable of playing only one melody drum, or "program." A more sophisticated example is the player piano, with its interchangeable paper rolls that operate on the same principle.

Still, there's something missing from a programmable player piano that keeps it from qualifying as a true computer. Even some of today's programmable calculators lack the essential element of computer intelligence. They, too, can be programmed to carry out a series of steps, but they can't imitate the decision-making power of a real brain.

What do computers have that all these other machines don't? *Conditional logic*. Although some other devices are capable of conditional logic on a very primitive level, no machine can do it as flexibly as a computer.

Michael Jackson Vs. Beethoven

Here's how conditional logic works. Let's say you send a friend to a record store with a $10 bill and these instructions: "If the store has Michael Jackson's latest album, then buy it for me. Otherwise, buy the Cleveland Symphony Orchestra's new recording of Beethoven's Fifth."

Now, you've done more than simply programmed your friend to visit the store and buy you a record. You've given him the power to make a decision in your absence, and also the information he needs to make the decision. Depending on whether a certain *condition* is met (if the store has Michael Jackson's latest album or not), your friend will act on either of the two alternatives (he'll buy you the Jackson record or the Beethoven record). Even if your friend has the brains of a hamster, he'll appear semi-intelligent to the record store clerk as he flips through the bins and picks the correct album.

All computer programming languages have commands that let you tell the computer to make the same sort of decisions. In BASIC, the most common command for conditional logic is the IF-THEN statement. It takes this form:

IF ***condition is met*** **THEN** ***perform this action.***

Computers don't understand English, of course, so the italicized words above must be replaced with terms the computer *can* understand. Usually the conditional part of the statement involves a comparison between variables and numbers. And often the resulting action will be a second command which sends the computer to another section of the program. Let's try some actual examples.

Conversational Computing

At one time or another you've probably used a computer program which carries on a conversation with you, depending on how you respond to certain questions. The machine appears almost human, and conditional logic is the key.

Let's say you're an insurance salesman who is writing a program designed to analyze a client's life insurance needs. One of the first questions the program needs to ask is the person's age. This can be done with a simple PRINT statement. Clear the computer's memory by turning it off, then on again, and type the following line exactly as it appears (remember that to enter a program line into memory, you must press the RETURN or ENTER key after typing the line):

```
10 PRINT "What is your age";
```

When the program runs, the PRINT statement will print the text between the quotes on the screen. Next, type this line:

```
20 INPUT A
```

When the program runs, this simple statement does three things. First, it prints a question mark after the text in the PRINT statement. Second, it makes the computer pause until someone types in a number on the keyboard. Finally, it takes the number and stores it in a memory location which can be referenced with the variable name A.

Now it's time for some conditional logic. Enter these two lines:

```
30 IF A>100 THEN PRINT "It's a little late to be
thinking about life insurance, isn't it?":END
40 PRINT "We have just the policy that you need."
```

Now clear the screen and run the program. When it asks for your age, try typing in a number less than 100. Then run the program again and type in a number greater than 100. See the difference? (Note: This program requires Extended BASIC on the TI-99/4A.)

Here's how it works. Line 30 compares the value stored in the variable A with 100. If A is greater than 100 (> is the *greater-than* sign), then the condition is met and the computer performs the instruction which immediately follows. *The second half of an IF-THEN statement is executed only when the condition in the first half is true.* (Incidentally, line 30 is also a *multistatement line;* a colon separates the second statement, END, from the IF-THEN statement. This command ends the program after the remark is printed.)

If the user claims to be less than 100 years old, the condition in line 30 is not met. Therefore, the computer ignores the rest of line 30 and continues (or "falls through") to the next line. The computer prints a different message on the screen and, presumably, would go on to the remainder of the program.

Simulated Intelligence

Although this simple four-line program contains only one conditional statement, it illustrates how computers can appear intelligent. The computer not only makes different responses depending on the user's input, it also seems to know that centenarians are unlikely candidates for life insurance. It even reveals a snappy sense of humor. Of course, it's really the programmer talking, not the computer. (Remember this the next time your computer makes a rude remark.)

Simple conditional statements like the one above are the basis for nearly all of what passes as artificial intelligence today. All programs work on the same principle, from the sophisticated modeling software that forecasts the nation's economy to the chess program which threatens the supremacy of the world's top champions. In fact, the chess program published in the December 1984 issue of COMPUTE!, when playing on level 5, uses a similar technique to evaluate up to 50 million possible moves during each turn. Naturally, a program that complicated must be written in machine language, not BASIC, or you'd be waiting months for the computer's response.

The IF THEN statement isn't the only way to simulate intelligence in BASIC. Most BASICs have at least two other statements which accomplish more or less the same thing. This indicates how important conditional logic really is in programming. (Linguists say you can determine a language's most often used concepts by counting the synonyms—interestingly, somebody once figured out that English has more terms for being inebriated than almost any other concept.)

In BASIC, as we mentioned, IF-THEN is by far the most common conditional statement. Some of the more powerful BASICs—such as IBM BASIC—augment the IF-THEN statement with an ELSE condition. This lets you combine two lines into one. For instance, lines 30 and 40 above could be rewritten like this in IBM BASIC:

```
30 IF A<100 THEN PRINT "We have just the policy
for your needs." ELSE PRINT "It's a little late to be
thinking about life insurance, isn't it?":END
```

ELSE doesn't let you do anything you couldn't do otherwise; it just makes programming more convenient.

Other examples of conditional statements in BASIC are ON-GOTO and ON-GOSUB. In effect, these let you combine a whole series of IF-THENs into one compact line. They are also called *conditional branching* statements because they branch to other parts of the program. An IF-THEN statement can be used for conditional branching, too. We'll cover both conditional and unconditional branching in next month's column, and also show a couple of ways to avoid long, cumbersome lists of IF-THENs in your programs.

Questions Beginners Ask

Q What exactly is a *crash* or a *freeze* and how can I avoid it? All I know so far is

C. Regena

Programming Without A Math Background

"Computer literacy," a required class in many high schools and colleges, is often little more than a class in elementary programming. Programming, however, is really only a small part of computing. Equally odd is the fact that many of these computer literacy classes require courses in algebra, calculus, or some other form of advanced mathematics as a prerequisite. In what way would knowing the calculus help someone learn BASIC?

Why are so many young people (often younger than 15) good programmers even if they've never taken algebra? Clearly, advanced mathematics has little to do with programming.

Of course, you do need to know a little about numbers. You need to know how to count. In a BASIC program the lines are numbered, so you must know the order in which the lines will be executed. Nevertheless, if you think logically, you can even use NUM to automatically number your lines as you are typing and you won't even have to worry about the line numbers.

If you like to program graphics, you should also learn something about basic coordinate geometry. That's just a mathematical term for using a grid. There are 24 rows and 32 columns on a TI-99/4A screen. If you want to place a character in a certain position, you have to tell the computer which row and column.

You'll also encounter numbers in the form of codes. For example, each color on a TI is given a number from 1 to 16. In any CALL SCREEN or CALL COLOR statement where you need a color number, you can look on the color chart to see which number represents which color. There are also codes for color sets, sounds, and characters. But beyond that, the most basic knowledge of addition, subtraction, multiplication, and division will be all you'll need in most cases.

Using Numbers Efficiently

Some skill at recognizing number patterns will help make your programs more efficient. Remember, however, that as long as your program works, it is "correct." There are many ways to accomplish the same task.

For instance, if you can recognize a pattern in your programming statements or among the numbers, quite often you can reduce the number of statements required. Suppose you want to draw seven horizontal lines across the screen. The lines are to be in rows 4, 7, 10, 13, 16, 19, and 22. You could use seven CALL HCHAR statements. Notice, though, that the numbers are each separated by 3. If you start with row 4 and add 3 each time until you get to 22, you'll have the lines you want. A FOR-NEXT loop could draw these same lines in only three statements:

```
200 FOR ROW=4 TO 22 STEP 3
210 CALL HCHAR(ROW,1,95,32)
220 NEXT ROW
```

Here's another problem. Suppose you want to draw a flower in several places on the screen, and each flower takes five characters, two on top of three others. The flowers are scattered randomly, so there's no pattern to their placement. In this case, a subroutine to draw the flower would be appropriate. Before you enter the subroutine, you could specify the row and column positions in the variables R and C. In the subroutine, the CALL HCHAR statements (or CALL VCHAR) need to be expressed in terms of R and C. If the upper-left corner of the flower is in position R,C then the next square would be R,C+1. Below R,C is R+1,C and next to it would be R+1,C+1 then R+1,C+2. The subroutine would look like this:

```
500 CALL HCHAR(R,C,112)
510 CALL HCHAR(R,C+1,113)
520 CALL HCHAR(R+1,C,114)
530 CALL HCHAR(R+1,C+1,115)
540 CALL HCHAR(R+1,C+2,116)
550 RETURN
```

And each time you need a flower, you would call the subroutine like this:

```
700 R=3
710 C=8
720 GOSUB 500
```

Streamlining Your Code

Now let's say you're drawing snakes instead of flowers. The snake still takes five characters, but all in a horizontal line. The subroutine might look like this:

```
500 CALL HCHAR(R,C,112)
510 CALL HCHAR(R,C+1,113)
520 CALL HCHAR(R,C+2,114)
530 CALL HCHAR(R,C+3,115)
540 CALL HCHAR(R,C+4,116)
550 RETURN
```

Notice that there is a pattern among the numbers. In each statement the column number increases by 1 and so does the character number. The five CALL HCHAR statements can be changed to:

```
500 FOR A=0 TO 4
510 CALL HCHAR(R,C+A,112+A)
520 NEXT A
```

A young friend came to me with a program in which he was randomly choosing five words, then printing them on rows 5, 7, 9, 11, and 13. He had to keep track of the words, their placement (order), and the answers. One solution was to DIMension arrays of W$ and ANS$ where the element specified was also the row number—so he had W$(R) and ANS$(R), where R was 5, 7, 9, 11, and 13. This method is easy to understand and worked well, but we were running into memory problems. Those arrays were taking up space because we weren't really using all the elements.

Notice that there is a pattern to the numbers:

Word 1—Row 5
Word 2—Row 7
Word 3—Row 9
Word 4—Row 11
Word 5—Row 13

The row numbers increase by 2. If you multiply each word number by 2, they become 2, 4, 6, 8, 10. Now compare these numbers with 5, 7, 9, 11, 13. Each of the word numbers (multiplied by 2) is 3 less than the row numbers. Therefore, if we have a word number N, the row number would be 2*N+3.

Later in the program, if we know the row number R and want to find the word number, we need to relate 5, 7, 9, 11, 13 to 1, 2, 3, 4, 5. First subtract 3 from the row number, then notice that the result is 2 times the word number. Given the row number R, the word number is (R−3)/2.

Quite often, if you line up a group of numbers you can see a relationship or a pattern. You can usually use standard arithmetic operations to get from one column of numbers to the next. If there is a progression of numbers, you can use a FOR-NEXT loop with a certain STEP size to get the right series of numbers.

Programming A Reflection

In this month's example program, we'll see how numbers can be manipulated to simulate reflections in graphics. The program takes a design you draw in the upper-left quadrant of the screen and creates reflections in the other three quadrants. We don't want the pattern simply repeated (as in the "Quilt Squares" program), rather, we want to actually reverse the image.

First, you draw a design in an area defined by rows 2 through 12 and columns 6 through 16. For example, the drawing starts in row 12 and column 16. This particular square reflects onto the other quadrants in squares (12,17), (13,16), and (13,17). The top-left square of the drawing quadrant is (2,6), or row 2 and column 6. The corresponding squares in the other quadrants are (2,27), (23,6), and (23,27).

In general, for a certain row R and column C, the corresponding square in the upper-right quadrant would be on the same row R and the column number would be 17 (the quadrant starts in the seventeenth column) plus (16−C). The first quadrant ends in column 16, and you subtract the first quadrant's column number to get its distance from the center. The result is 33−C. Another way to look at it is that the column number will be the same distance from the last column as the original square is from the first column—thus 32−C+1 or 33−C.

The corresponding square in the lower-left quadrant will have the same column number C as the original square, but the row will be 12+13−R or 24+R−1, which is 25−R. The lower-right quadrant has the same row as the lower-left quadrant and the same column as the upper-right quadrant. Thus the three corresponding squares are (R,33−C) and (25−R,C) and (25−R,33−C). Lines 620–640 and 1000–1020 use these relationships.

Electronic Snowflake

When I was a child I liked to fold paper, cut a design, then unfold the paper to see what it looked like. Sometimes we would fold the paper to get a six-sided snowflake. Other times we would fanfold the paper. We also used different variations of simply folding the paper into rectangles.

This "Snowflake" program is the computerized version of cutting paper snowflakes (with no scraps of paper to clean up). Suppose you have a square piece of paper. Fold it in half to make a rectangle, then fold the rectangle in half to make a square. Now cut a design in that square. Unfold the paper and you have a four-sided snowflake.

When you run this program, you will see a large square outlined. You can draw in the upper-left square only. Use the arrow keys to move the cursor, press F to fill the cursor position with color, and press the space bar to preserve the background color (or to erase a previously filled position). When your design is complete, press ENTER. The computer starts at the center and moves outward to reflect your pattern on the other quadrants of the larger square.

When the design is complete, you can press M to modify, S to start a new pattern, P to print the pattern if you have a printer, and ENTER to end the program. If you press M to modify, the cursor starts blinking again and you can resume drawing. But this time your changes appear immediately in the rest of the design. When you're finished, you can press ENTER again. If you press S to start a new pattern, the screen clears and you can start over.

To use the printer option, the printer must be attached and switched on (don't forget the RS-232 interface). Line 800 contains the printer configurations; modify it if necessary. The hard copy printout is elongated but shows the pattern you drew. Filled squares are represented by asterisks and the blanks by dots. If you want, you could even use this pattern for counted cross-stitching or needlepoint.

Program Explanation

Lines 110–200 clear the screen and print the title and instructions. Lines 210–270 define characters used as graphics. Characters 96–99 are used to outline the large square and the drawing quadrant. Character 104 is the filled square, and character 105 is the cursor used in drawing. Character 112 is the yellow dot used to indicate the ENTER key after the snowflake is complete. Lines 280–290 define the colors for the snowflake and the ENTER key symbol. If you wish to use different colors for the snowflake, change the color number 5 in line 280 and the screen color in line 430.

Lines 300–390 wait for you to press ENTER, then continue the instructions. Lines 400–410 wait for you to press any key to start. Lines 420–490 clear the screen, change the screen color to cyan (light blue), then outline the large square and the upper-right quadrant.

Lines 500–510 define the starting row X and column Y for the drawing cursor. Lines 520–540 call the subroutine that is the procedure for moving and filling in squares until the ENTER key is pressed.

When you press ENTER, lines 550–570 make a beeping sound, then erase the lines for the quadrant. Lines 580–660 look at each square in the upper-right quadrant. If they find a filled square, they draw a square in the other quadrants in the corresponding position. This happens in loops, starting with the center square and moving outward (by columns C) and upward (by rows R). When the process is complete, line 670 sounds another beep.

Lines 680–730 print the options to press M for modify, S to start over, P to print, or ENTER to end. Lines 740–780 detect the key pressed and branch accordingly.

Lines 790–920 contain the printing option. You must have a printer connected, and your printer configuration must be specified in line 800. The computer looks at each row from 2 to 23 and each column from 6 to 27 using CALL GCHAR, and then prints a period for a space and an asterisk for a filled square. After the printing is complete, the program branches back to the options of M, S, P, and ENTER.

Lines 930–1030 contain the modify option. First the options at the right of the square are cleared. Then the drawing cursor reappears. Design changes instantly appear in the other three quadrants. When you press ENTER, the program branches back to the options of M, S, P, and ENTER.

Lines 1040–1320 contain the subroutine for the drawing procedure. CALL GCHAR checks to see what character is in position X,Y and calls that character number (G). Lines 1060–1080 blink the cursor while waiting for a keypress. Lines 1090–1300 are the branching statements executed when certain keys are pressed. Line 1310 draws the new character if it is a space or a filled square.

Lines 1330–1340 clear the screen, then end the program.

If you wish to save typing, you can receive a copy of this program by sending a blank cassette or disk, a stamped, self-addressed mailer, and $3 to:

C. Regena
P.O. Box 1502
Cedar City, UT 84720

Please be sure to specify that you need the TI version of Snowflake.

Snowflake

Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
100 REM  SNOWFLAKE
110 CALL CLEAR
120 PRINT TAB(9);"SNOWFLAKE":::
130 PRINT "USE THE ARROW KEYS TO DR
    AW"
140 PRINT :"IN THE UPPER LEFT QUADR
    ANT."
150 PRINT :"PRESS 'F' TO FILL A SQU
    ARE."
160 PRINT :"PRESS SPACE BAR TO ERAS
    E."
170 PRINT :"PRESS <ENTER> WHEN YOU"
180 PRINT :"ARE FINISHED DRAWING."
190 PRINT ::"THE COMPUTER WILL COMP
    LETE"
200 PRINT :"THE SNOWFLAKE."
210 CALL CHAR(96,"ØØØØØØØØØØØØØØFF")
220 CALL CHAR(97,"8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø")
230 CALL CHAR(98,"FF")
240 CALL CHAR(99,"Ø1Ø1Ø1Ø1Ø1Ø1Ø1Ø1")
250 CALL CHAR(1Ø4,"FFFFFFFFFFFFFFFF
    ")
260 CALL CHAR(1Ø5,"FF818181818181FF
    ")
270 CALL CHAR(112,"3C7EFFFFFFFF7E3C
    ")
280 CALL COLOR(1Ø,5,1)
290 CALL COLOR(11,12,1)
300 PRINT ::"PRESS <ENTER>."
310 CALL KEY(Ø,K,S)
320 IF K<>13 THEN 31Ø
330 CALL CLEAR
340 PRINT "AFTER SNOWFLAKE IS COMPL
    ETE,"
350 PRINT :"PRESS <M> TO MODIFY PAT
    TERN"
360 PRINT :"PRESS <S> TO START OVER"
370 PRINT :"PRESS <P> TO PRINT COPY"
380 PRINT :"PRESS <ENTER> TO END."
390 PRINT ::::"PRESS ANY KEY NOW T
    O START."
400 CALL KEY(Ø,K,S)
410 IF S<1 THEN 4ØØ
420 CALL CLEAR
430 CALL SCREEN(8)
440 CALL HCHAR(1,6,96,22)
450 CALL VCHAR(2,28,97,22)
460 CALL HCHAR(24,6,98,22)
470 CALL VCHAR(2,5,99,22)
480 CALL VCHAR(2,17,97,11)
490 CALL HCHAR(13,6,98,11)
500 X=12
510 Y=16
520 CALL SOUND(15Ø,1397,2)
530 GOSUB 1Ø5Ø
540 IF K<>13 THEN 53Ø
550 CALL SOUND(1ØØ,1497,2)
560 CALL HCHAR(13,6,32,11)
570 CALL VCHAR(2,17,32,11)
580 FOR R=12 TO 2 STEP -1
590 FOR C=16 TO 6 STEP -1
600 CALL GCHAR(R,C,H)
610 IF H=32 THEN 65Ø
620 CALL HCHAR(R,33 C,H)
630 CALL HCHAR(25-R,C,H)
640 CALL HCHAR(25-R,33-C,H)
650 NEXT C
660 NEXT R
670 CALL SOUND(1ØØ,44Ø,2)
680 CALL VCHAR(8,29,6Ø,4)
690 CALL VCHAR(8,31,62,4)
700 CALL HCHAR(8,3Ø,77)
710 CALL HCHAR(9,3Ø,83)
720 CALL HCHAR(1Ø,3Ø,8Ø)
730 CALL HCHAR(11,3Ø,112)
740 CALL KEY(Ø,K,S)
750 IF S<1 THEN 74Ø
760 IF K=83 THEN 42Ø
770 IF K=13 THEN 133Ø
780 IF K<>8Ø THEN 93Ø
790 REM  PRINTER CONFIGURATION
800 OPEN #1:"RS232.BA=6ØØ"
810 FOR R=2 TO 23
820 FOR C=6 TO 27
830 CALL GCHAR(R,C,H)
840 IF H<>32 THEN 87Ø
850 PRINT #1:".";
860 GOTO 88Ø
870 PRINT #1:"*";
880 NEXT C
890 PRINT #1:CHR$(13)
900 NEXT R
910 CLOSE #1
920 GOTO 67Ø
930 IF K<>77 THEN 74Ø
940 CALL VCHAR(8,29,32,4)
950 CALL VCHAR(8,3Ø,32,4)
960 CALL VCHAR(8,31,32,4)
970 CALL SOUND(15Ø,1397,2)
980 GOSUB 1Ø5Ø
990 IF K=13 THEN 67Ø
1000 CALL HCHAR(X,33-Y,G1)
1010 CALL HCHAR(25-X,Y,G1)
1020 CALL HCHAR(25-X,33-Y,G1)
1030 GOTO 97Ø
1040 REM  SUB TO DRAW
1050 CALL GCHAR(X,Y,G)
1060 CALL KEY(Ø,K,S)
1070 CALL HCHAR(X,Y,1Ø5)
1080 CALL HCHAR(X,Y,G)
1090 IF K=13 THEN 132Ø
1100 IF K=7Ø THEN 13ØØ
1110 IF K=32 THEN 128Ø
1120 IF K<>88 THEN 116Ø
1130 IF X=12 THEN 1Ø6Ø
1140 X-X+1
1150 GOTO 1Ø5Ø
1160 IF K<>83 THEN 12ØØ
1170 IF Y=6 THEN 1Ø6Ø
1180 Y=Y-1
1190 GOTO 1Ø5Ø
1200 IF K<>68 THEN 124Ø
1210 IF Y=16 THEN 1Ø6Ø
1220 Y=Y+1
1230 GOTO 1Ø5Ø
1240 IF K<>69 THEN 1Ø6Ø
1250 IF X=2 THEN 1Ø6Ø
1260 X=X-1
1270 GOTO 1Ø5Ø
1280 G1=32
1290 GOTO 131Ø
1300 G1=1Ø4
1310 CALL HCHAR(X,Y,G1)
1320 RETURN
1330 CALL CLEAR
1340 END
```

Commodore 64 and Atari computers); *Gulf Strike,* a simulation of land, air, and naval combat in the Middle East ($30 disk, for Atari computers); and *Clear for Action,* a ship-to-ship combat game ($25 cassette, $30 disk, Atari computers).

Microcomputer Games, Inc.
The Avalon Hill Game Co.
4517 Harford Rd.
Baltimore, MD 21214

Apple II Word Processor

The *Milliken Word Processor,* a program designed to teach children the fundamentals of writing on a computer, has been released for Apple II series computers by Milliken Publishing Company.

It teaches how to use the computer for composing, structuring, editing, and filing written material, and has most basic word-processing functions, including graphics to ease understanding of functions.

Designed for children ages seven and older, the word processor retails for $69.95.

Milliken Publishing Co.
1100 Research Blvd.
P.O. Box 21579
St. Louis, MO 63132

New Text Adventures

Infocom has released two new text adventures for most home computers, *Suspect* and *The Hitchhiker's Guide to the Galaxy.*

In *Suspect,* the player takes the role of a newspaper reporter invited to a masquerade ball—who ends up being accused of murder. You must prove your innocence, and also find out who committed the crime and why.

The Hitchhiker's Guide to the Galaxy is an adaptation of the novel of the same name by Douglas Adams. The player is protagonist Arthur Dent, who goes off on a journey through the universe with his friend, Ford Prefect.

Suspect retails for $39.95 for the Atari and Commodore 64 versions, and $44.95 for versions on most other personal computers. *Hitchhiker's Guide to the Galaxy* has a suggested retail price of $34.95 for the Commodore and Atari versions, with other versions retailing for $39.95.

Infocom, Inc.
55 Wheeler St.
Cambridge, MA 02138

Bridge For IBM, Apple, Commodore

A bridge game for one or more players, *BridgePro,* has been released for the IBM PC and PCjr, Apple II series, and Commodore 64 computers by Computer Management Corporation.

BridgePro allows one person to bid with all hands randomly dealt. Other options are two-player versions, a best hand option, replay of hands, and separate instructions for beginning bridge players.

Suggested retail price is $35. *BridgePro* is available on disk.

Computer Management Corporation
2424 Exbourne Ct.
Walnut Creek, CA 94596

Critical Thinking Program

MaxThink, a program for the IBM PC and compatibles which provides commands for high-level thinking processes such as analysis, synthesis, and evaluation, has been introduced by MaxThink, Inc.

The software uses commands for organizing, analyzing, evaluating, planning, and thinking about information.

The suggested retail price is $60. It requires 192K of memory.

MaxThink, Inc.
230 Crocker Ave.
Piedmont, CA 94610

Personal Productivity Software

Arrays, Inc./Continental Software has introduced several personal productivity software packages for Apple and IBM computers, including:

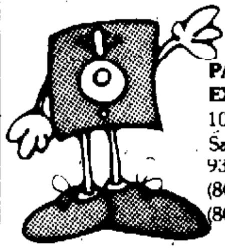
The Home Accountant Expanded, an accounting package for Apple IIc and IIe computers ($74.95 suggested retail price); and a new, compiled version of *The Home Accountant Plus* ($149.95) for the IBM PC.

Also, educational versions of *Ultra-File,* a filing, reporting, and graphics package; *Property Management,* a program to record transaction history for residential and/or commercial income property-related charges;

MEMOREX
FLEXIBLE DISCS

WE WILL NOT BE UNDERSOLD!! Call Free **(800)235-4137** for prices and information. Dealer inquiries invited and C.O.D.'s accepted.

VISA



PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA 93401. In Cal. call (800)592-5935 or (805)543-1037