The Joy Of Joysticks: 12 Joysticks Compared

COMPUTE!

$2.50
February
1983
Issue 33
Vol. 5, No. 2
63379 £1.85 in UK

The Leading Magazine Of Home, Educational, And Recreational Computing

All New Games Issue

How The Pros Write Computer Games

***Slalom:* 3-D Simulation Game For Atari And PET**

***A Day At The Races:* Game With Excellent Graphics And Sound Tutorial For VIC And Atari**

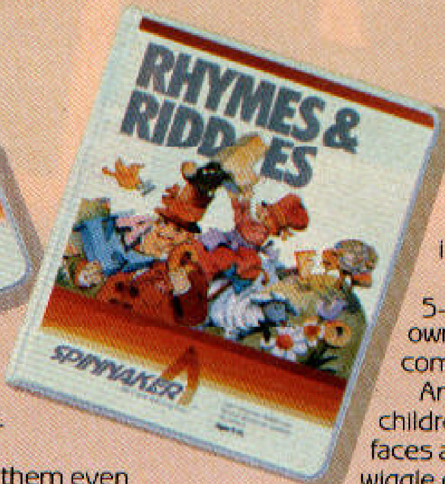
Super Shell Sort For PET/CBM

Creating Graphics On An Expanded VIC

SuperFont Plus For The Atari

Plus Game Reviews VIC, Atari, Sinclair/Timex, Apple, And More

Spinnaker's early learning games will help make your children as smart as you tell everyone they are.



Your kids are pretty smart. After all, they're *your* kids.

Spinnaker can help make them even smarter. With a line of educational software that kids love to play.

Spinnaker games make the computer screen come to life with full color graphics and sound. And they're fun. Lots of fun. But they also have real educational value.

Some of our games help exercise your child's creativity. Others improve memory and concentration. While others help to improve your child's writing, vocabulary, and spelling skills.

And every Spinnaker game provides familiarity with the computer and helps your children feel friendly with the computer. Even if they've never used a computer before.

And Spinnaker games are compatible with the most popular computers: Apple,® Atari® and IBM.®

Our newest game, KinderComp™ (Ages 3-8) is a collection of learning exercises presented in a fun and exciting manner.

Rhymes and Riddles™ (Ages 4-9) is a letter guessing game featuring kids' favorite riddles, famous sayings and nursery rhymes.

Story Machine™ (Ages 5-9) lets children write their own stories and see them come to life on the screen.

And FACEMAKER™ lets your children create their own funny faces and make them wink, smile, wiggle ears (not your kids' ears, the ears on the screen), etc.

And we're introducing new games all the time.

So look for Spinnaker games at your local software retailer, or by writing to: Spinnaker Software, 215 First St., Cambridge, MA 02142. And show your kids how smart their parents really are.

SPINNAKER™

We make learning fun.

February 1983 Vol. 5, No. 2

NOTE: See page 248 before typing in programs.

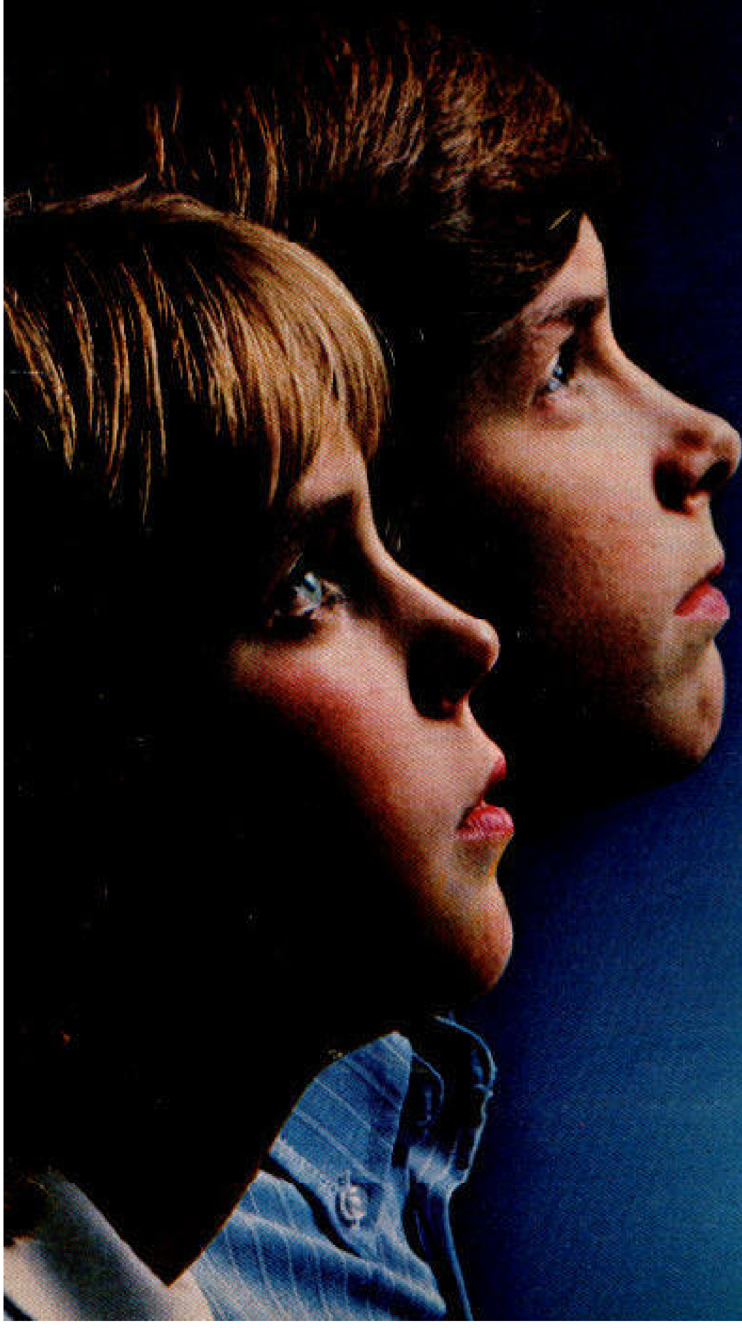
GUIDE TO ARTICLES AND PROGRAMS

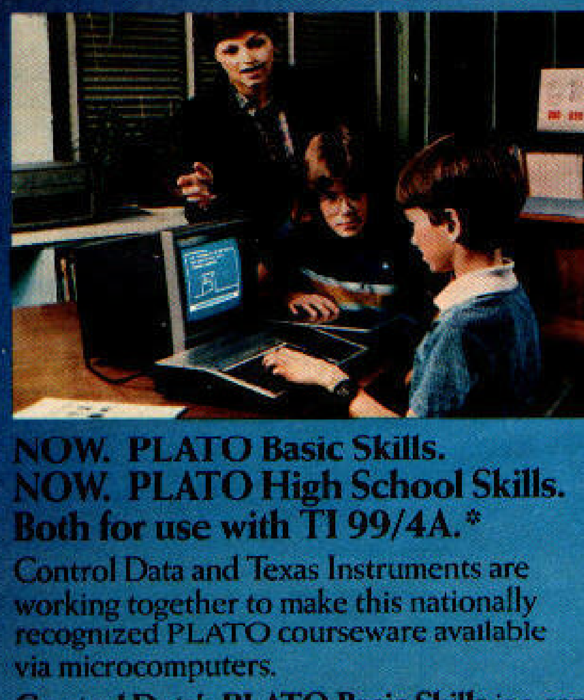
AP Apple, **AT** Atari, **P** PET/CBM, **V** VIC-20, **O** OSI, **C** Radio Shack Color Computer, **64** Commodore 64, **ZX** Sinclair ZX-81, * All or several of the above.

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published 12 times each year by Small System Services, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919)275-9809. Editorial Offices are located at 625 Fulton Street, Greensboro, NC 27403. Domestic Subscriptions: 12 issues, $20.00. Send subscription orders or change of address (P.O. form 3579) to Circulation Dept., **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403. Second class postage paid at Greensboro, NC 27403 and additional mailing offices. ISSN 0194-357X.

PLATO WIDENS YOUR CHILD'S WORLD.

Announcing new educational courseware for microcomputers to improve your child's Basic Skills, High School Skills or Foreign Language vocabulary.





NOW. PLATO Basic Skills.
NOW. PLATO High School Skills.
Both for use with TI 99/4A.*

Control Data and Texas Instruments are working together to make this nationally recognized PLATO courseware available via microcomputers.

Control Data's PLATO Basic Skills is a comprehensive curriculum designed to enhance the development of elementary and junior high Reading, Math and Grammar skills.

Control Data's PLATO High School Skills help high school level students master Reading, English, Math, Social Studies and Science. It has been used to assist students preparing for G.E.D. exams.

*For Basic and High School Skills pricing and order information, write: Texas Instruments, Education Department, Box 53, Lubbock, TX 79408.

NEW!* Nine lessons in Foreign Languages for use with Apple II Plus.

Three lessons each for French, Spanish and German. Each lesson uses a hangman or pyramid game to help children learn words most associated with defined activities. For junior or senior high students.

Travel Words: Children study French, Spanish or German words they need to know to buy train, plane or bus tickets; order in a restaurant; request medical assistance; read street signs; etc.

Shopping Words: Children study French, Spanish or German words they need to know to shop for food, clothing, etc.

Classroom Words: Children study the French, Spanish or German words that are common to the school environment.

*Available March 30

Lessons Available For Apple II Plus, TI 99/4A and Atari 800.

Basic Number Facts: Practice in addition without carrying; subtraction without borrowing; and multiplication/division with single digits. For elementary students.

Whole Numbers: Practice in addition, subtraction, multiplication, division and mixed numbers. For elementary and junior high students.

Decimals: Practice locating decimal numbers on the number line. For elementary students.

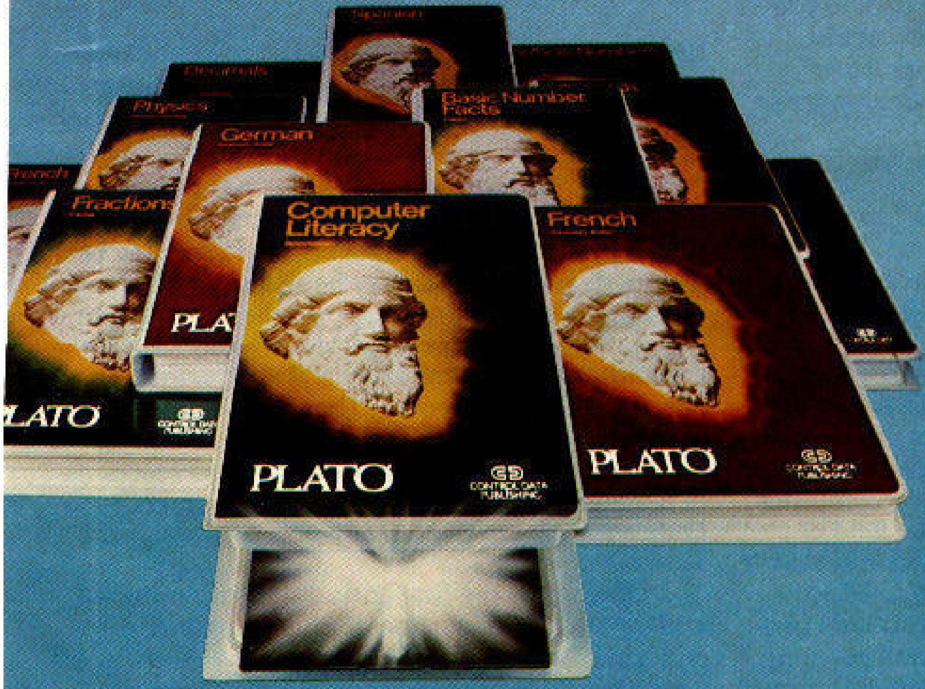
Fractions: Same skill level and format as decimals.

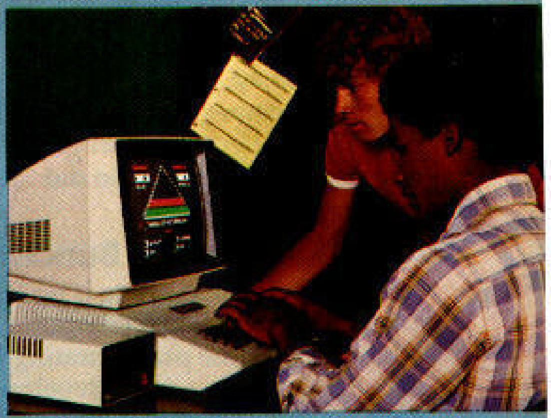
Physics—Elementary Mechanics: Students are shown a physical problem, then must "purchase" the missing information to answer it correctly. For senior high physics students.

French, German, Spanish Vocabulary Builders: Gives children a basic vocabulary of 500 words. Supplements introductory and refresher courses.

Computer Literacy—Introduction: For jr. or sr. high and vocational school students.

INTRODUCTORY OFFERING:
- Single lesson, $45.00
- Additional lessons, $35.00 ea.
- Additional disk included with each lesson ordered at no extra charge
- 10 day money-back trial





SEND FOR PLATO COURSEWARE CATALOG

For a free copy of our PLATO catalog, or to order, mail reply card; call toll-free 800/233-3784; or write Control Data Publishing Co., P.O. Box 261127, San Diego, CA 92126. In California, call 800/233-3785.

CONTROL DATA PUBLISHING

PLATO®

COMPUTER-BASED EDUCATION

WIDEN YOUR CHILD'S WORLD WITH THIS INTRODUCTORY OFFERING.

10 DAY MONEY-BACK GUARANTEE.

Now, until May 31, 1983, when ordering a lesson at $45.00, you may order additional lessons for just $35.00 each. (Additional disk is included with each lesson at no additional charge.)

Mail this form, or call toll-free 800/233-3784. (In California, call 800/233-3785.)

Apple® is a trademark of Apple Computer, Inc.
Atari® is a registered trademark of Atari, Inc.
TI® is a registered trademark of Texas Instruments.

Warranty available free from the Control Data Publishing Co., 4455 Eastgate Mall, San Diego, CA 92121.

BEFORE ORDERING: CHECK TO MAKE SURE YOUR EQUIPMENT MEETS THESE REQUIREMENTS.

	Apple II PLUS	Atari 800	TI 99/4A
Memory:	48K	48K	48K
No. of drives required:	1 disk and controller	1 disk and controller	1 disk and controller
Operating System:	D.O.S. 3.3	D.O.S. 2	PLATO Interpreter Cartridge*

Will display on any color or b/w monitor or TV compatible with your microcomputer listed.

*PLATO Interpreter Cartridge may be ordered directly with the order form below. Only one per TI 99/4A machine is needed.

CONTROL DATA PUBLISHING CO., P.O. Box 261127, San Diego, CA 92126

SHIP TO: (Please Print)

Name______________________________

Address______________________________

City________________ State______ Zip______

☐ Payment enclosed (Check or money order only. Make payable to Control Data Publishing Co.).

☐ Mastercharge ☐ VISA ☐ American Express ☐ Diners Club

Card Number ________________ Expiration Date______

Your signature________________ Phone____________

INTRODUCTORY PRICES: One lesson $45.00. Each additional lesson $35.00.

Price for order: . . . . $______
Calif. Residents: add 6% sales tax. . . . . $______
Add $2 shipping and handling. . . . . $ 2.00
TOTAL PRICE: . . . . $______

☐ Include PLATO Interpreter Cartridge for my TI 99/4A. Price $50.00.

☐ Please send information on ____________ lesson(s).

☐ Please send ________ copies of your courseware catalog.

Please complete the following: (Indicate computer for which you are ordering courseware.)

APPLE II PLUS ☐ ATARI 800 ☐ TI 99/4A ☐

Quantity	Title (see course description for skill level)
	Basic Number Facts
	Whole Numbers
	Decimals
	Fractions
	Physics—Elementary Mechanics
	French Vocabulary Builder
	German Vocabulary Builder
	Spanish Vocabulary Builder
	Computer Literacy Introduction

Allow 3-5 weeks for delivery

Available on APPLE II PLUS only

Quantity	Title (see course description for skill level)
	French—Travel Words
	German—Travel Words
	Spanish—Travel Words
	French—Shopping Words
	German—Shopping Words
	Spanish—Shopping Words
	French—Classroom Words
	German—Classroom Words
	Spanish—Classroom Words

Available March 30. Allow 3-5 weeks for delivery

If not completely satisfied, you may return all lesson material within 10 days of receipt for a refund. All orders subject to acceptance. This introductory offering expires May 31, 1983.

Control Data Publishing Company supplies this courseware under a personal license agreement rather than for sale. Customers obtain the right to possess and use PLATO courseware by paying the prices specified and agreeing to the terms and conditions of the license agreement contained in the package.

The game that puts space games in perspective. Zaxxon™, one of the most popular arcade games of 1982, is now available for use with your home computer system.

Zaxxon™ technology and creativity present a 3-dimensional-like playfield which sets Zaxxon™ apart from other computer games.

Zaxxon™ looks and sounds like aircraft flight, and players can soar to new levels of home computer entertainment. From the daring attack on the enemy's floating fortress and the blazing battle against the enemy's fighter fleet to the final showdown with the deadly armored robot, Zaxxon™ challenges the skill and imagination of every player at every level of skill.

Imagine yourself the pilot, attacking the enemy fortress–climbing, diving, strafing to score points and extra fuel. The enemy fights back with a barrage of missiles and gunfire. Then you face a fleet of enemy fighters in a gripping dogfight of altitude strategy and flying skill. Survive this battle and the enemy's fortress, defended with laser barriers, then you've earned the ultimate challenge; a blazing confrontation with the powerful robot, armed with a lethal homing missile.

Zaxxon™ is the one game that you must see to believe. You have to play it to feel its impact. If you're ready to face the challenge, check with your local software dealer or send check or money order with $2.00 postage/handling. California residents add 6½% sales tax. Available on cassette or diskette. Suggested retail price $39.95.

Available in January on Atari®, February on Apple® and Radio Shack® Color, and April on TI 99/4A™ and NEC 6000™.

Datasoft Inc.®
COMPUTER SOFTWARE
9421 Winnetka Avenue
Chatsworth, CA 91311
(213) 701-5161

Part II:

Writing Transportable BASIC

Edward Ordman

This concludes a two-part article on writing BASIC programs so that they are more easily read, revised, or translated to run on different computer brands. Though not everyone will agree with the goal (general-purpose BASIC), or the approach (structured programming), many of these suggestions are potentially useful to those programmers who later revise and improve their own programs. For contrast, see the views of some of the programmers quoted in "How The Pros Write Computer Games," elsewhere in this issue.

Structure

The major tool in making a program transportable is careful attention to program structure. This does not mean slavish adherence to "structured programming." It does mean using common sense and some of the important tools available to keep programs from becoming "spaghetti bowls" of GOTOs. This can include "structured programming" when applicable.

To consider a concrete example, suppose we have two branches in our code governed by a GOTO. A simple version might be:

```
500 IF X>2 THEN T=T+Y:C=C+2 ELSE T=T+Z:
    C=C+1
```

There is certainly no objection to writing this in one line if your BASIC allows it; the intent is clear. Remember that you should leave space for new lines, since someone may have to rewrite this as:

```
500 IF X>2 THEN 504
501 T = T+Y
502 C = C+2
503 GOTO 507
504 T = T+Z
505 C = C+1
507 REM ENDIF
```

Even this is still quite readable. It is clear where the IF starts and where its effect ends. A far worse example (but painfully common in beginners' programs) would have IF X>2 THEN 4000 and then down at line 4000 would have:

```
4000 T=T+Z: C=C+1: GOTO 510
```

This is hard to read: how, when checking line 4000, can you know where it relates to the rest of the program? Reading lines 500-510, how can you understand the options of the other path?

My own practice, incidentally, is to avoid GOTOs over long distances, avoid upward GOTOs unless they are part of a fairly formal structure, and have GOTOs go to REM statements in a great many cases. Suppose, in the example above, line 510 was PRINT TAB(C);X;TAB(C+5); Y and some variation in the new machine meant that this had to be expanded to two lines to get the right spacing. A GOTO 510 in line 503 means that a line 509 cannot be introduced without other changes; the 507 REM means changes in the PRINT do not require changes in the IF.

A similar situation arises in programs where there is a large loop (PLAY AGAIN in a game) and some initialization before it. If you start

```
1 PRINT "WELCOME TO THE GAME"
2 T = 0
3 X = RND(1)
4 Y = 1
```

the person rewriting this may type 2 T=0: X=RND(1): Y=1, and be in big trouble when he discovers that at line 5560 you have GOTO 4. He will be in more trouble when he revises the program and needs to add another statement within the main loop, but before Y=1. Compare the program:

```
10 PRINT "WELCOME TO THE GAME"
20 T = 0:X = RND(1):REM INITIALIZE,0<
   X<1
30 REM  ENTER MAIN LOOP HERE
40 Y = 1 :REM  COUNT NUMBER OF ATTEMP
   TS
5560 GOTO 30 :REM REPEAT MAIN LOOP
```

In this version, the rewriter will not confuse line 20 and line 40; a line 35 can be added; and there is no confusion as to exactly where the GOTO is leading, even after several program revisions. In general, do not GOTO "the middle" of a line of reasoning without clearly labeling why and providing an easy way to make changes without extensive rewriting.

If you really want to avoid upward GOTOs in as many cases as possible (and it *does* make programs easier to read!), there are two alternative structures that are important: GOSUB ... RETURN and the DO ... WHILE. First, let us consider the DO...WHILE.

DO...WHILE can be regarded as an extension of FOR...NEXT. A typical form is:

```
1000 DO WHILE X>10
1010 PRINT X
1020 T = T+X
1030 X = X/2
1040 ENDWHILE
```

Suppose X is 50 when this is entered. Lines 1010-1030 will be done for X=50, for X=25, for X=12.5; then X will become 6.25, the test will fail, and the program will go on after line 1040. This is a remarkably useful *thinking* tool, even if your BASIC does not have these statements (many do not). But, for transportability, I would argue *against* using these statements even if you have them. There is, however, no reason at all not to *think* in terms of DO...WHILE and then to write an imitation of it:

```
1000 IF X<=10 THEN 1040 :REM DO WHILE
     X>10 TO LINE 1040
1010 PRINT X
1020 T = T+X
1030 X = X/2
1035 GOTO 1000
1040 REM  END WHILE
```

Again, this is easy to read, the upward GOTO is clearly explained, and the reader is in no doubt as to the scope of the loop and where you enter and leave it.

Subroutines Are Best

Subroutines – the facility provided by GOSUB and RETURN – are the single most important feature in providing transportability. There is a strong case to be made for dividing every program of more than a few dozen lines, and many shorter ones, into subroutines. Ideally, each subroutine should have a purpose that you can describe in one or two lines, and that explanation should be given in remarks at the head of the subroutine. The subroutine should *not* interact with the rest of the program except as provided in the leading remarks. An example:

```
6000 REM  SUBROUTINE TO CONVERT TO PO
     LAR COORDINATES
6001 REM  GIVEN X,Y COORDINATES.  RET
     URN R=RADIUS,T=ANGLE.
6002 REM  X,Y UNCHANGED.  RETURN T=0
     IF R=0.
6003 REM
6010 R=SQR(X*X + Y*Y)
6020 IF R = 0 THEN T = 0 : RETURN
6030 IF X<>0 THEN T = ATN(Y/X) : RETU
     RN : REM ARCTANGENT, RADIANS
6040 IF Y > 0 THEN T = 3.14159/2
6050 IF Y < 0 THEN T = -3.14159/2
6090 RETURN
```

It is entirely appropriate for subroutines to call other subroutines, or for a main program to consist primarily of subroutine calls, with all the real work done in the subroutines. But when this is done, it is even more important to make sure that the subroutines can be debugged separately – that they do not, for instance, change the variable used elsewhere, but not mentioned in the headnote.

Where you are using a feature that you know is particular to your computer – for instance, disk input/output – it is especially important to isolate it in a subroutine, and label it as machine-dependent. This means that it can be rewritten later with a minimum of change to the main program logic.

Make Input/Output General

It is very likely that anyone rewriting a program for another machine will have to revise input/output statements. This applies to PRINT and INPUT for keyboards, terminals, CRTs, and printers; to cassette and disk storage; to game controllers and joysticks; and to all other peripherals. Essentially the only "minimal" features that all machines have in common are INPUT X and PRINT X,Y,Z, and even these are not as standard as one might like. The usual solution is to stick to minimal formatting, if you consider transportability of prime importance; or to place fancy input/output in subroutines and indicate your intention clearly, if it is essential to the program. Here we can give only a quick guide to some of the tricks and pitfalls.

INPUT Some computers allow you to cue the user (prompt) as desired, e.g., INPUT "YOUR NEXT GUESS?";N while others do not. The others can fake it by PRINT "YOUR NEXT GUESS";:INPUT N getting the question mark on the same line as the printout. Many BASICs will not allow suppression of the question mark. Inputting string variables, particularly with embedded spaces or commas, also differs dramatically from system to system, as mentioned earlier. If your program depends heavily on a precise form of string input, place the input routine in a subroutine and explain the purpose carefully. For example:

```
2000 REM  STRING S$ WILL BE ALL CHARA
     CTERS TYPED (PRINTABLE OR NOT)
2001 REM  UNTIL ENTER IS HIT (EXCLUDI
     NG THE ENTER)
2010 S$ = ""
2020 K$ = INKEY$ :REM  GETS SINGLE KE
     Y FROM KEYBOARD
2030 IF K$ = "" THEN 2020
2040 IF ASC(K$) = 13 THEN 2090 :REM
     CARRIAGE RETURN, OR ENTER
2050 S$ = S$ + K$
2060 GOTO 2020
2090 RETURN
```

Of course, other machines may require substantial rewriting of this subroutine, if the special word INKEY$ is not available or works differently. In some microcomputers, the implementation may be as easy as INPUT LINE S$. Still, having this in a single subroutine, rather than scattered throughout the program, will simplify the job of rewriting for a new machine.

PRINT Some computers allow statements like PRINT "$"X"000", without commas or semicolons, and produce the output $4000 when X is 4. Others require PRINT "$";X;"000" and produce $ 4 000 or something similar. Usually, a

WHAT'S SMALLER THAN A BUSINESS CARD?

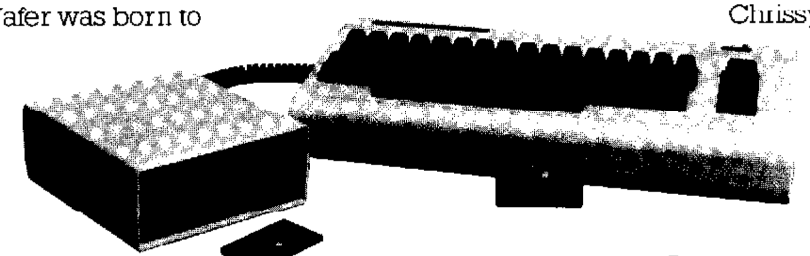
FASTER THAN CASSETTES?

AND FAR LESS EXPENSIVE THAN DISKS?

Why the ESF-20/64 Stringy Floppy from Exatron, of course. Our exciting little storage alternative gives you near floppy disk speed and reliability at a budget-minded price. Our high quality digital mass storage system is the perfect product to fill the gap between cassettes and floppy disks. And that's especially true in the case of the Commodore VIC-20® and Commodore 64® microcomputers.

The Exatron Stringy Floppy system is based on a small endless loop tape cartridge we call "The Wafer". This wafer measures only 2 3/4" x 1" x 3/16" — or about the dimensions of a standard business card. Wafer was born to run fast — at around 7200 baud, or 14 times faster than a standard cassette and has a memory capacity of 64K bytes. The ESF-20/64 system costs less than $200 and wafers are less than $3.00 each. No wonder computer people call us the "poor man's disk system".

If you'd like to get the world on a string, and bring your VIC-20® or Commodore 64® into the 20th Century, just fill out the coupon below and mail it to EXATRON, 181 Commercial Street, Sunnyvale, CA 94086. If you need to place an order call (408) 737-7111, outside California (800) 538-8559 and ask for Chrissy or Natalie.



The ESF-20/64 Stringy Floppy System from Exatron

for use with the Commodore VIC-20® and Commodore 64® microcomputer.

exatron

For exciting alternatives in mass storage.

Name ______________________

Address ______________________

City/State/Zip ______________________

Mail today for information on the ESF-20/64 from EXATRON, 181 Commercial St., Sunnyvale, CA 94086 clear indication (in a REMark) of what you want is far more helpful than an ingenious trick to achieve it on your machine. The exact meanings of comma and semicolon differ from one machine to another: it is universal that comma means "wide space; arrange in columns" and semicolon means "short space, or no space," but the details differ. In many configurations, TAB will not work properly (this is common when using a printer attached to an Apple, for instance).

If you must engage in any fancier spacing than use of commas and semicolons, explain yourself in REMarks and leave it to the reader to implement it on his machine. Many microcomputers do not have PRINT USING; if you use it, an example output line contained in a remark is very helpful. If you use a fancy PRINT statement repeatedly in your program, consider placing it to a subroutine where the reader will have to translate it only once.

CLEAR There are a number of special commands whose implementation differs from one computer to another. Some examples are Clear Screen, Go to top-of-page, and similar ones. (Varying print character width, for instance, is usually a function of the printer model, not of the BASIC.) If at all possible, place these functions on a line by themselves and remark clearly; it will then be easy for the reader to translate them, or delete them if inapplicable to the new system.

Joysticks These also differ dramatically from one system to another. Again, place them in a clearly labeled section of the program, preferably a subroutine, and label what they do. In particular, avoid repeating these statements numerous places within the program. Example:

```
1050 GOSUB 5000 : REM  READ PADDLES
2300 GOSUB 5000
5000 REM  READ PADDLES X,Y -- VALUES
     ARE 0 - 255, SCALE TO 0 - 100
5010 X = PDL(0)/2.55
5020 Y = PDL(1)/2.55
5030 RETURN
```

Clearly, someone whose paddle-reading commands are different, or give values in a different range, can easily rewrite this subroutine.

Tape/Disk While the particular statements involved in tape and disk input/output differ for almost every system, the general functions to be performed are almost identical. Typically, one must specify a file name and number by which it will be referred, and whether it will be for input (READ or INPUT), or output (WRITE or PRINT), or both. A typical statement is something like OPEN "DATAFILE" AS 1 FOR INPUT. If your BASIC allows omitting some of this, include it in a REMark. For example,

```
1050 PRINT D$; "OPEN INPUT";F$ :REM OPEN F$,
     SEQUENTIAL, INPUT ONLY
```

is acceptable if you only have one file open at a time; the reader can insert an AS #1 if the new system requires it. Once you have opened a file, you must read from it or write to it, typically by a statement such as READ #1, A,B,C or PRINT #3,A;C$;B;C$;X :REM C$ = ",".

Notice that if your system does not require a specific indication in the statement that it refers to a file, you should include one in a REMark. It is an excellent idea to write commas as field dividers to a file, even if your system will permit a space as a divider on input. Enough systems insist on the comma that it decreases portability to omit it. A statement such as

```
1060 REM A TYPICAL LINE OF FILE IS 4 , 5 ,
     DEBITS , 2.95 (CR)
```

will often make the program much clearer to the reader than it is from just the line

```
INPUT#3 P(K),Z(K),D$,A(I)
```

In the case of a direct access file, most systems also need to know the record length and record number for each read or write. If a direct access file is opened for updating, you should read a record before you write it. Finally, on any type of file, you should remember to close it explicitly (usually CLOSE #3 or some variant). Even if your BASIC does not insist on this, someone else's will; and it can be hard to figure out *when* to do the closing in a strange program.

A program using no files is more easily transportable than one using files; the fewer the files, the more transportable. (Avoid opening more files than needed at one time.) Sequential files are easier to move than direct access files; files read or written "all-at-once" are more transportable than ones that are read or written only intermittently. If at all possible, structure a program like this:

```
1000 GOSUB 7500 : REM READ WHOLE FILE
      INTO AN ARRAY
 .... :REM  MAIN PROGRAM ACTS ON THE
ARRAY
4000 GOSUB 7700 : REM WRITE WHOLE ARR
     AY BACK OUT TO FILE
4010 GOTO 9999
```

so that all file-handling is confined to specific subroutines and the files can be kept on a cassette tape even without fancy automatic stop-start features.

Graphics

If we view BASIC as something almost geological, something that has had layers added over time, graphics capabilities are the last layer, and the layer least solidified. Graphics differ more from machine to machine than any other feature. Fancy graphics tricks are the very hardest thing to transport from one system to another. Still, it is possible to do some graphics work and still limit the problems when moving them to another system.

Generally, it is easiest to transport a program that uses only "character" graphics. If we view the screen as consisting of a fixed number of rows and a fixed number of columns, then each position can be occupied by one letter or "character." If we confine ourselves to commonly available characters, our program *should* be capable of being rewritten for most systems. If it does not involve moving pictures, it should even be possible to run it on a printer-oriented system in many cases.

As you know, common systems *do* differ in screen size (in number of characters in a row or column). The first thing we must do is let the reader know what assumptions we have made:

```
50 M1=16 :REM NUMBER OF LINES ON SCREEN
60 M2=40 :REM NUMBER OF CHARACTERS PER
   LINE
```

From this point on, we should place everything in terms of the numbers M1 and M2, *not* 16 and 40. Further, to position a given character C$ at coordinates X,Y (that is, X across and Y down: position X of row Y), we should set X, Y, and C$ and then call a subroutine. On the IBM Personal Computer, we print an "A" in the center of the screen by

```
100 X=INT(M2/2) :Y=INT(M1/2) :C$="A"
110 GOSUB 7000
 ....
7000 REM  SUBROUTINE TO WRITE C$ AT
     POSITION X OVER, Y DOWN ******
7010 LOCATE Y,X : PRINT C$;
7020 RETURN
```

Again, the user of any given computer can rewrite this subroutine as a whole far more easily than he can rewrite statements like LOCATE 12,40: PRINT "A"; which are scattered throughout the program.

Sometimes a screen is built up by "jumping around," rather than line-by-line. If you wish to get hard copy of such a screen, and lack a built-in operating system procedure to do so, you can have the subroutine just mentioned build an array by 7015 S(X,Y) = ASC(C$) (or 7015 S$(X,Y) = C$) and later print the entire array. This may be as easy as:

```
8000 REM  PRINT THE SCREEN STORED IN
     ARRAY S(M2,M1) *************
8010 FOR I = 1 TO M2
8020 FOR J = 1 TO M1
8030 PRINT CHR$(S(J,I)); :REM ; LEAVE
     S NO SPACE ON IBM PERSONAL COMP
8040 NEXT J
8050 PRINT :REM  GO TO NEXT LINE - DE
     LETE IF IT CAUSES DOUBLE SPACING
8060 NEXT I
8070 RETURN
```

Note that this program must contain a line such as

```
70 DIM S(80,24) :REM SAVE SCREEN. NOTE DIM
   S(M2,M1)
```

so that a person changing M1 and M2 will know how it changes the DIM statement.

A remarkable assortment of graphics effects may be achieved just by the skillful use of standard characters: minus signs or underscores for horizontal lines, ones or a special symbol for verticals, and so on. It is not hard to generate pictures by hand: hold a piece of window screen over a picture, judge the amount of darkness as best you can (most people can rate "darkest, dark, middle, light, clear") and use characters such as M I : . and space to represent them. Some scaling may be needed; in many systems the space allocated for a character is 1 2/3 to 2 times as tall as it is wide. Fill-in-the-blanks effects, on screen or paper, may be achieved by using minus signs as underscores:

SOCIAL SECURITY NUMBER ? ---·--·----

Turning now to "high-resolution" graphics, or other extended graphics features, we find that most of them still can be expressed in terms of X-Y coordinates and making a specific mark at specific coordinates, although the mark is now usually "on" or "off" or "COLOR 7" instead of a letter. The same principle as before applies: specify the maximum size involved; if at all possible give dimensions as fractions of M1 and M2 rather than absolute numbers; and keep the actual writing in as few subroutines as possible.

In general, have one subroutine that draws a point; another that draws a line by making repeated GOSUBs to the subroutine to mark points; and so on. Even if your computer has built-in line-drawing commands, place them in subroutines (instead of HLIN 20,50 TO 30,40 write X1 = 50: Y1 = 20: X2 = 40: Y2 = 30: GOSUB 2600 where 2600 has the line HLIN Y1,X1 TO Y2,X2), so that a person whose computer lacks them can try to write a reasonable imitation.

If you write carelessly, or depend too heavily on features of a particular machine, you can have a program that is very hard to translate to any other machine. If you want to be able to move your programs to a new, different machine, or have them run on a friend's machine or on a machine at school, you must plan ahead when you first write the program.

It takes relatively little extra effort to write a transportable program, and there are many fringe benefits. You yourself will find the program easier to test, debug, or reread a few months later. A little avoidance of particular machine "special features," a little use of good structuring practices, and some care to isolate likely-to-change features in labeled subroutines, can pay off in far easier maintenance and rewriting. And if it means that some published programs will run on a larger variety of machines than they used to, it will pay off for all of us.

Stake a Claim on the most exciting New Game for your Atari Home Computer.
MINER 2049ER
STARRING BOUNTY BOB
BY BILL HOGUE
• From the programming team that brought you "Robot Attack", "Defense Command" and many other great Arcade games for your TRS-80™
• 100% machine language
• 16K ROM Cartridge, the largest available anywhere!
• Written specifically for the Atari® — not a converted Apple® game.
• Ten different rounds
• Difficulty adjustment
• High score table
• Demo mode
• Spectacular sound and graphics
• Runs on any 400/800 with at least 16K memory
• Only $49.95
Here are just three of ten rounds in the game:
Round 1: The Mine Shaft.
Sharpen your miner's skill!
Round 3: The Transporters.
Beam yourself up!
Round 10: The Cannon.
Shoot yourself to the top!
BIG FIVE SOFTWARE
P.O. Box 9078-185 · Van Nuys, CA 91409 · (213) 782-6861
To order, see your local dealer. If they do not have "Miner 2049er" then send a check or money order to us for $49.95 plus $2.00 for shipping and handling for immediate shipment! (California residents add 6½% sales tax.)
"Atari," "Apple," and "TRS-80" are trademarks of Atari Inc., Apple Computer, and Tandy Corp., respectively.

Writing Effective Educational Programs

C. Regena

When using the computer as a "teacher," you should consider several factors which are unique to this relatively new situation: how people best learn from machines. Computer tutorial programming techniques are illustrated with a geometry-teaching program for the TI-99/4A, the VIC, and the TRS-80 Color Computer (with 16K Extended BASIC).

One of the most natural uses for a microcomputer is in education. A student may use a tutorial (teaching) program to learn at his or her own pace, or use a drill program to get practice and experience.

Two capabilities of the computer are useful in educational programs. First, a computer does not get tired of repetition. A teacher or parent may get frustrated or not have time for many repetitions, but a computer has as much time to run the program as the student wishes. Second, the randomness feature can be used to change numbers each time a drill is performed or to mix up the order of questioning or to individualize instruction and practice.

Successful Tutorials

Either in programs you purchase or in programs you write yourself, several attributes should be incorporated.

Color graphics. Just a lot of words on a screen are hard to read and tiring. The program should not mimic a book. Graphics can be used appropriately to illustrate the concepts being taught.

Music, sound, and speech. Music can add variety and enhancement to a program to retain interest. Speech can be effective in reading, spelling, language programs, or in any programs for young children who may not yet be proficient in reading. Keep in mind the educational concept that the more senses the student uses (sight, sound, touch), the more efficient the learning process.

Positive and negative reinforcement. A short musical interlude, or perhaps a change in graphics, may be used for correct answers. A non-intimidating "uh-oh" tone or noise may be used for incorrect responses. Be careful that the incorrect answer doesn't result in an overly entertaining display, or the student will *want* to get the wrong answer. Avoid name-calling and "smart remarks" that are intended to be cute, but actually detract from educational programs.

Remediation. After correct answers the program should advance to higher levels of difficulty or to new concepts. After an incorrect response or two, the correct answer should be presented. Usually with a true-false or yes-no question, the student wouldn't need to be told the answer, but after an input answer which could be one of many answers, the answer is necessary.

Flexibility. The student should be able to advance quickly over sections she or he already knows and to repeat sections as needed (use menu screens or options). Also, any time the student needs to read something, she or he should be able to pause as long or as short a time as desired. It is frustrating to be reading when the program changes screens before you're ready – or to have to spend a certain length of time with a screen that you are already familiar with.

Careful use of INPUT. Keep in mind that any time a student needs to respond to an INPUT there is a greater chance of the program "crashing" or of graphics getting messed up. After an INPUT, be sure to check allowable limits. What happens if a string variable is entered when a numeric variable is expected, or vice versa? If you can arrange questions or choices to require a one-key-press response, your program will be easier for the student to use and have fewer chances for error.

If you must use INPUT, make sure the student knows what is expected, and ask for only one response at a time. Usually in scientific or higher mathematics programs you can assume the student will know what type of number is expected, but in elementary or beginning programs the student must be guided.

Plotting Points

"Coordinate Geometry" is a tutorial program written for the TI-99/4A, TRS-80 Color Computer (16K and Extended BASIC), and VIC-20 that teaches how to locate points on a rectangular coordinate grid. The program includes a section for positive and negative coordinates.

First a random example point is given with the coordinates labeled. Next a random point is given, and the student must press the numbers for the x-coordinate and the y-coordinate. The third step is to locate the point, given the coordinates. The TI-99/4A and TRS-80 CC have standard arrow keys. As an arrow key is pressed, the point moves in that direction. For the VIC-20 I chose to use the function keys since there are no standard arrow keys. F1 is up, F3 is left, F5 is right, and F7 is down.

To detect which key is pressed, the TI-99/4A uses CALL KEY (0,K,S), where K is the ASCII code of the key pressed. K is checked for (up), (left), D (right), and X (down), and any other key pressed is ignored. (Lines 1420-1690)

On the TRS-80 CC, INKEY$ is used to detect a key pressed. The character codes for the arrow keys are checked for the point to move. (Lines 1020-1180)

The VIC-20 needs the GET function to determine the key pressed. (Lines 57-72)

These programs use the graphics capabilities of the computers to illustrate the grid. A PRINT statement is used for the graphics because it is quicker than a series of CALL HCHAR or CALL VCHAR statements (TI-99/4A), SET commands (TRS 80 CC) or POKE commands (VIC-20). The grid is drawn several times in the program, so the instructions to draw it are in a subroutine.

A musical arpeggio is played for a correct answer, and an "uh-oh" is played for an incorrect answer. These procedures are also in subroutines and may be called from several places in the program.

After an incorrect answer, the correct answer is given. The student can study the problem, then press ENTER or RETURN, and another problem will be given. Numbers are chosen randomly. If the answer is correct, the student has the choice of another problem of the same type or of continuing the program.

Only key presses are necessary in the TRS-80 CC and VIC-20 versions and the first section of the TI-99/4A version. Later sections of the TI-99/4A program require INPUT for positive and negative coordinates and answers which may require a decimal.

If you wish to save typing time and effort and would like a copy of any of these programs, you may send $3, a blank cassette, and a self-addressed, stamped mailer. Be sure to specify which computer version.

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

Explanation of the Program: TI-99/4A

Line Numbers	
100	Defines random function.
110-460	Print title screen and define graphics characters.
490-510	Subroutine for incorrect answer music.
520-560	Subroutine for correct answer music.
570-610	Subroutine to print grid.
620-660	Subroutine to print "PRESS ENTER" and wait for response.
670-710	Subroutine to draw graphics.
720-870	Give random example of a point with coordinates.
880-910	Print instructions.
920-1240	Exercise for giving coordinates for a point.
1250-1280	Print instructions.
1290-1840	Exercise for locating a point with given coordinates.
1850-1900	Subroutine to randomly choose point.
1910-1940	Subroutine to draw vertical red line from point to x-axis.
1950-1980	Subroutine to draw horizontal red line from point to y-axis.

Explanation of the Program: TRS-80 CC

Line Numbers	
50	Branch to title screen.
60-180	Subroutine to print grid.
190-220	Subroutine to print "PRESS ENTER" and wait for response.
230-270	Subroutine to choose point and calculate graphics print position.
280-310	Subroutine to calculate coordinates to SET point.
320	Subroutine to play music for incorrect answer.
330	Subroutine to play music for correct response.
340-540	Draw title screen.
550-600	Define string variables for grid graphics; pause.
610-710	Draw grid, show example point.
720-900	Present problem to find coordinates for given point.
910-960	Print instructions.
970-1270	Present problem to locate point with given coordinates.
1280-1340	Print choice to have another problem, start over, or end program; branch appropriately.
1350	End.

Explanation of the Program: VIC-20

Line Numbers	
2	Prints title screen.
4	Defines volume and sound.
6-7	Define string variables for grid; delay.
8-26	Draw grid; show example point.
30-51	Present problem to give coordinates for given point.
52-54	Print instructions.
55-74	Present problem to locate point with given coordinates.
75-79	Print choice to have another problem, start over, or end program, branch appropriately.
80-83	Subroutine to label point and draw yellow lines from point to axes.
84	Subroutine to calculate graphics memory location.
86	Subroutine to play music for incorrect answer.
88	Subroutine for correct answer.
89	Subroutine to delay for music.
90-92	Subroutine to print grid.
94	Subroutine to get rid of buffered keys in GET function.
96-99	Subroutine to print "PRESS RETURN" and wait for response.
100	End.

Program 1: TI-99/4A Version

```
100 DEF R(N)=INT(N*RND+1)
```

```
110 CALL CLEAR
120 PRINT"   ************************":"   *";TAB
    (25);"*":"  * COORDINATE GEOMETRY  *"

130 PRINT "  *";TAB(25);"*":"  *****************
    ********"::::TAB(11);"POINTS":::
140 A$="1818181818181818"
150 B$="181818FFFF181818"
160 C$-"000000FFFF"
170 FOR C=96 TO 112 STEP 8
180 CALL CHAR(C,A$)
190 CALL CHAR(C+1,B$)
200 CALL CHAR(C+2,C$)
210 NEXT C
220 CALL CHAR(120,"183C7EFFFF7E3C18")
230 CALL CHAR(128,"183C7EFFFF7E3C18")
240 CALL CHAR(129,"00000000030C30C")
250 CALL CHAR(130,"030C30C")
260 CALL CHAR(64,"3C4299A1A199423C")
270 CALL CHAR(94,"00102828444482FE")
280 CALL COLOR(10,5,1)
290 CALL COLOR(11,10,1)
300 CALL COLOR(12,11,1)
310 CALL COLOR(13,7,1)
320 CALL CHAR(140,"1010101010101010")
330 CALL CHAR(141,"000000FF")
340 CALL CHAR(142,"101010F")
350 CALL COLOR(14,13,1)
360 A$="`  h  h  h  h  h  h  h"
370 B$="ajjijjijjijjijjijjji"
380 C$="abbabbabbabbabbabbabb"
440 CALL CLEAR
450 CALL COLOR(2,2,1)
460 GOTO 720
490 CALL SOUND(100,330,2)
500 CALL SOUND(100,262,2)
510 RETURN
520 CALL SOUND(100,262,2)
530 CALL SOUND(100,330,2)
540 CALL SOUND(100,392,2)
550 CALL SOUND(200,523,2)
560 RETURN
570 CALL CLEAR
580 PRINT "    Y":"    ";A$:"    ";A$:"   4";B
    $:"    ";A$:"    ";A$:"   3";B$
585 PRINT "    ";A$:"    ";A$:"   2";B$
590 PRINT "    ";A$:"    ";A$:"   1";B$:"    "
    ;A$:"    ";A$:"   0";C$
595 PRINT "    0  1  2  3  4  5  6  7":::
600 CALL HCHAR(20,31,88)
610 RETURN
620 PRINT TAB(16);"PRESS<ENTER>";
630 CALL KEY(0,K,S)
640 IF K<>13 THEN 630
650 CALL HCHAR(24,18,32,13)
660 RETURN
720 GOSUB 570
730 PRINT "THE LOCATION OF A POINT IS":"GIVEN
    BY ITS X-COORDINATE"
735 PRINT "AND Y-COORDINATE (X,Y)"
740 RANDOMIZE
750 X=R(5)
760 GOSUB 1850
770 GOSUB 1910
780 CALL HCHAR(Y1,X1+2,40)
790 CALL HCHAR(Y1,X1+3,48+X)
800 CALL HCHAR(Y1,X1+4,44)
810 GOSUB 1950
820 CALL HCHAR(Y1,X1+5,48+Y)
830 CALL HCHAR(Y1,X1+6,41)
840 PRINT :"WANT ANOTHER EXAMPLE? (Y/N)";
850 CALL KEY(0,K,S)
860 IF K=89 THEN 720
870 IF K<>78 THEN 850
880 CALL CLEAR
890 PRINT "YOU WILL BE SHOWN A POINT."::"PRESS
    THE NUMBER OF THE"
895 PRINT :"X-COORDINATE THEN THE"
900 PRINT :"NUMBER OF THE Y-COORDINATE.":::::
    :
910 GOSUB 620
920 CALL CLEAR
930 GOSUB 570
940 PRINT :::
950 RANDOMIZE
960 GOSUB 1850
970 CALL HCHAR(21,7,40)
980 CALL HCHAR(21,9,44)
990 CALL HCHAR(21,11,41)
1000 CALL KEY(0,K,S)
1010 CALL HCHAR(21,8,63)
1020 CALL HCHAR(21,8,32)
1030 IF S<1 THEN 1000
1040 CALL HCHAR(21,8,K)
1050 X2=K
1060 CALL KEY(0,K,S)
1070 CALL HCHAR(21,10,63)
1080 CALL HCHAR(21,10,32)
1090 IF S<1 THEN 1060
1100 CALL HCHAR(21,10,K)
1110 Y2=K
1120 IF X2<>X+48 THEN 1190
1130 IF Y2<>Y+48 THEN 1190
1140 GOSUB 520
1150 PRINT "PRESS":"1 FOR SAME TYPE PROBLEM":"2
     TO CONTINUE PROGRAM";
1160 CALL KEY(0,K,S)
1170 IF K=49 THEN 920
1180 IF K=50 THEN 1250 ELSE 1160
1190 GOSUB 490
1200 GOSUB 1910
1210 GOSUB 1950
1220 PRINT "THE CORRECT ANSWER IS (";STR$(X);",
     ";STR$(Y);")"
1230 GOSUB 620
1240 GOTO 920
1250 CALL CLEAR
1260 PRINT "NOW YOU WILL BE GIVEN THE"::"COORDI
     NATES."
1265 PRINT :"USE THE ARROW KEYS TO MOVE"::"THE
     POINT TO THE CORRECT"
1270 PRINT :"PLACE, THEN PRESS <ENTER>."::::
1280 GOSUB 620
1290 CALL CLEAR
1300 GOSUB 570
1310 RANDOMIZE
1320 X=R(7)
1330 Y=R(4)
1340 X1=7+3*X
1350 Y1=17-3*Y
1360 PRINT :"PLOT (";STR$(X);",";STR$(Y);")"::
1370 C1=97
1380 A-17
1390 A1=A
1400 B=7
1410 B1=B
1420 CALL HCHAR(A,B,120)
1430 CALL KEY(0,K,S)
1440 IF S<1 THEN 1430
1450 IF K=13 THEN 1700
1460 IF K<>69 THEN 1510
1470 IF A=5 THEN 1430
1480 CALL GCHAR(A-3,B,C)
1490 A=A-3
1500 GOTO 1650
1510 IF K<>88 THEN 1560
1520 IF A=17 THEN 1430
1530 CALL GCHAR(A+3,B,C)
1540 A=A+3
1550 GOTO 1650
1560 IF K<>83 THEN 1610
1570 IF B=7 THEN 1430
1580 CALL GCHAR(A,B-3,C)
1590 B=B-3
1600 GOTO 1650
1610 IF K<>68 THEN 1430
1620 IF B=28 THEN 1430
1630 CALL GCHAR(A,B+3,C)
1640 B=B+3
1650 CALL HCHAR(A1,B1,C1)
```

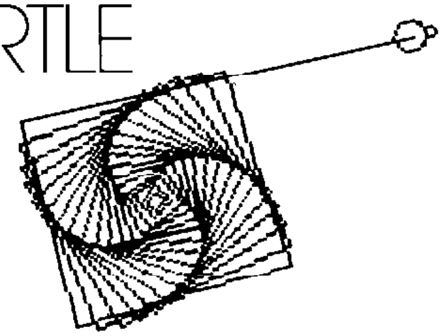
```
1660 A1=A
1670 B1=B
1680 C1=C
1690 GOTO 1420
1700 CALL SOUND(150,1397,2)
1710 CALL GCHAR(Y1,X1,C)
1720 IF C=120 THEN 1790
1730 GOSUB 490
1740 CALL HCHAR(Y1,X1,128)
1750 GOSUB 1910
1760 GOSUB 1950
1770 GOSUB 620
1780 GOTO 1290
1790 GOSUB 520
1800 PRINT "PRESS":"1 FOR SAME TYPE PROBLEM":"2
     TO CONTINUE PROGRAM";
1810 CALL KEY(0,K,S)
1820 IF K=49 THEN 1290
1830 IF K<>50 THEN 1810
1840 END
1850 X=R(7)
1860 Y=R(4)
1870 X1=7+3*X
1880 Y1=17-3*Y
1890 CALL HCHAR(Y1,X1,128)
1900 RETURN
1910 FOR I=Y1+1 TO 17
1920 CALL HCHAR(I,X1,112)
1930 NEXT I
1940 RETURN
1950 FOR I=X1-1 TO 7 STEP -1
1960 CALL HCHAR(Y1,I,114)
1970 NEXT I
1980 RETURN
```

Program 2: TRS-80 Color Computer Version

```
50 GOTO 340
60 CLS:PRINT@1,"Y";A$
70 PRINT@33,"3";B$
80 PRINT@66,A$
90 PRINT@98,A$
100 PRINT@129,"2";B$
110 PRINT@162,A$
120 PRINT@194,A$
130 PRINT@225,"1";B$
140 PRINT@258,A$
150 PRINT@290,A$
160 PRINT@321,"0";B$
170 PRINT@354,"0{4 SPACES}1
    {4 SPACES}2{4 SPACES}3{4 SPACES}4
    {4 SPACES}5  X"
180 PRINT:RETURN
190 PRINT @496,"PRESS <ENTER>";
200 R$=INKEY$:IF R$=""THEN 200
210 IF ASC(R$)<>13 THEN 200
220 RETURN
230 X=RND(5)
240 Y=RND(3)
250 A=322-96*Y+X*5
260 PRINT @A,CHR$(159);CHR$(159);
270 RETURN
280 B=4+X*10:C=20-6*Y
290 FOR I=C+2 TO 20:SET(B,I,4):SET(B+
    2,I,4):NEXT
300 FOR I=B-2 TO 4 STEP -1:SET(I,C,4)
    :NEXT
310 RETURN
320 PLAY "L16;O2;E;C":RETURN
330 PLAY "L16;O2;CEG;L8;O3;C":RETURN
340 PMODE 4,1:PCLS:SCREEN 1,0
350 DRAW "S8;BM20,65"
360 DRAW "NU6;R4;U1;BM+3,+1"
370 DRAW "BM+1,0;H1;U4;E1;R2;F1;D4;G1
    ;L2;BM+6,0"
380 DRAW "BM+1,-0;H1;U4;E1;R2;F1"
```

```
390 DRAW "BM+0,+4;G1;L2;BM+6,0"
400 DRAW "U4;E2;F2;D2;NL4;D2;BM+3,0"
410 DRAW "BM+2,+0;U6;NL2;R2;BM+3,+6"
420 DRAW "BM+1,0;R1;NR1;U6;NL1;R1;BM+
    4,+6"
430 DRAW "U6;F1;D1;F2;D1;F1;NU6;BM+3,
    0"
440 DRAW "BM+1,-0;H1;U4;E1;R2;F1"
450 DRAW "BM+0,+2;NL1;D2;G1;L2;BM+6,0
    "
460 DRAW "BM+7,0;U6;R3;F1;D1;G1;L3;BM
    +7,3"
470 DRAW "BM+1,0;H1;U4;E1;R2;F1;D4;G1
    ;L2;BM+6,0"
480 DRAW "BM+1,0;R1;NR1;U6;NL1;R1"
490 DRAW "BM+4,+6;U6;F1;D1;F2;D1;F1;N
    U6;BM+3,0"
500 DRAW "BM+2,+0;U6;NL2;R2;BM+3,6"
510 DRAW "BM+0,-1;F1;R2;E1;U1;H1;L2;H
    1;U1;E1;R2;F1"
520 FOR I=104 TO 168 STEP 16:LINE(20,
    I)-(236,I),PSET:NEXT
530 FOR I=28 TO 220 STEP 16:LINE(I,10
    0)-(I,172),PSET:NEXT
540 CIRCLE(76,136),4:CIRCLE(156,152),
    4
550 C$=CHR$(175)
560 D$=C$+C$+C$+C$+C$:E$="{3 SPACES}"
    +C$+C$
570 A$=E$:B$=D$
580 FORI=1TO4:A$=A$+E$:B$=B$+D$:NEXT
590 A$=C$+C$+A$:B$=C$+C$+B$+C$+C$
600 FOR D=1 TO 3000:NEXT
610 GOSUB 60
620 PRINT"A POINT HAS AN X-COORDINATE
    "
630 PRINT"AND A Y-COORDINATE (X,Y)."
640 X=RND(4):GOSUB 240
650 X$=RIGHT$(STR$(X),1):Y$=RIGHT$(ST
    R$(Y),1)
660 PRINT@A+2,"("+X$+","+Y$+")";
670 GOSUB 280
680 PRINT@480,"ANOTHER EXAMPLE? (Y/N)
    ";
690 R$=INKEY$:IF R$=""THEN 690
700 IF R$="Y" THEN 610
710 IF R$<>"N" THEN 690
720 GOSUB 60
730 GOSUB 230
740 PRINT@416,"WHAT ARE THE COORDINAT
    ES?"
750 PRINT"{5 SPACES}(?,?)"
760 X1$=INKEY$:IF X1$="" THEN 760
770 PRINT @454,X1$;
780 Y1$=INKEY$:IF Y1$="" THEN 780
790 PRINT @456,Y1$;
800 IF X<>VAL(X1$) THEN 820
810 IF Y=VAL(Y1$) THEN 850
820 GOSUB 320:GOSUB 280
830 PRINT@460,"LOCATION IS (";RIGHT$(
    STR$(X),1);",";RIGHT$(STR$(Y),1);
    ")"
840 GOSUB 190:GOTO 720
850 GOSUB 330:PRINT@460,"CORRECT!":GO
    SUB 190
860 PRINT@496,"{13 SPACES}";
870 PRINT @416,"PRESS 1 FOR SAME TYPE
     PROBLEM{9 SPACES}2 TO CONTINUE P
    ROGRAM"
880 R$=INKEY$:IF R$="" THEN 880
890 IF R$="1" THEN 720
900 IF R$<>"2" THEN 880
910 CLS
920 PRINT@66,"YOU WILL BE GIVEN THE":
    PRINT"  COORDINATES."
```

FRIENDS OF THE TURTLE



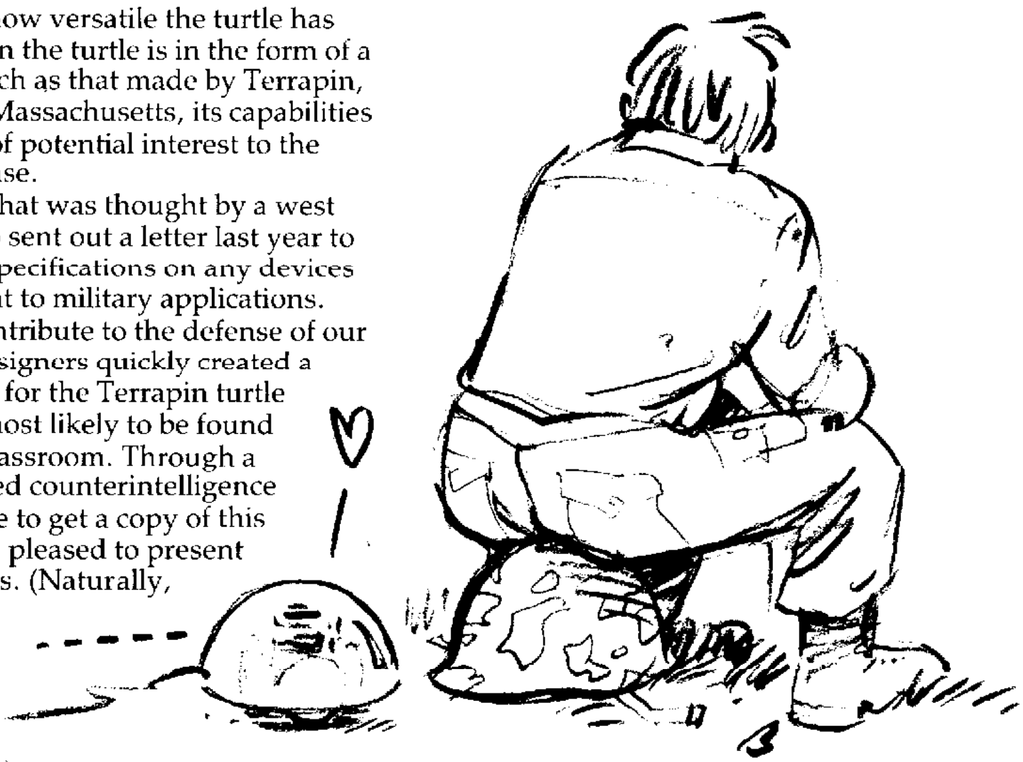
David D. Thornburg, Associate Editor

The Department Of Turtle Defense

Those of us who limit our use of turtle graphics to the aesthetic pleasures of art or to its use in education have no idea how versatile the turtle has become. In fact, when the turtle is in the form of a mechanical robot, such as that made by Terrapin, Inc., in Cambridge, Massachusetts, its capabilities are so great as to be of potential interest to the Department of Defense.

At least this is what was thought by a west coast think-tank who sent out a letter last year to Terrapin asking for specifications on any devices that might be relevant to military applications.

Ever eager to contribute to the defense of our country, Terrapin designers quickly created a military specification for the Terrapin turtle – a $600 peripheral most likely to be found in a primary grade classroom. Through a network of well placed counterintelligence operatives, I was able to get a copy of this specification and am pleased to present the following excerpts. (Naturally, I have made sure that I haven't included any information that would compromise our nation's defense.)



From a functional viewpoint, Turtles show great promise as all-terrain vehicles for pushing heavy payloads to their destination. Under the heading of survivability, we find that:

> *The turtle enjoys a low observability, due to a minimal radar cross section and an almost non-existent infrared signature. In addition, its ground-hugging characteristics maximize terrain masking, resulting in lower target acquisition by most classes of SSM and ASM threats. ... The Turtle can make a 180-degree turn in less space than any military vehicle currently in use by US forces, ground, air, or sea. With minor modifications, a Turtle could be constructed that could double its cruising speed for a terminal "dash" capability that would greatly enhance survivability in the endgame.*
>
> *... Even if a suitable counter were found to all these properties of the Turtle, it is doubtful that an enemy could afford to deploy counter-weapons in sufficient number to nullify the possibility of defense saturation in the event of an all-out Turtle attack.*

On the topic of range ...

> *The Mark I, Mod 0 Turtle has an effective range of some 3 to 4 meters, depending on its winding count. Range is most severely limited by the Turtle's cable, but this limitation is trivial by comparison to the inherent advantages of wire-guidance. ... Furthermore, our research department is currently engaged in the testing of a 100 mile cable for the Turtle. ... While this does result in a shorter tooth-to-tail ratio, we feel it could significantly enhance the battlefield capabilities of the Turtle installations.*

On the topic of guidance ...

> *Because the Terrapin Turtle™ is computer con-*

the player questions. In this way, the computer adds information to its knowledge base so it improves how well it plays the game.

There are two other programs. "Layer Cake" is a version of the *Tower of Hanoi* puzzle. It is easy to use and takes good advantage of the Apple's graphics capabilities. "Raise the Flags" is a non-violent variation of *Hangman*. The disk also contains a word editor which lets you enter your own words for the Raise the Flags program.

Instant Zoo is a disk for children ages seven to ten. Its four games are: Instant Zoo, a picture recognition game similar to Guess Who, but with animals instead of Muppets; Star Watch, which measures how long the child takes to press a key after a shooting star appears on the screen; Quick Match, in which two words are shown and the child presses one key if they match and another if they do not match; and Scramble, an anagram game in which the child races the computer to unscramble words. There is also a word editor for creating your own word lists for Quick Match and Scramble.

Spotlight is a disk for children ages nine to thirteen. It has two programs in which the child turns mirrors to direct lights to targets. The third program, called "Hot Stuff," is a game of logic in which the player tries to guess the computer's secret three-digit number. After each guess, the computer tells how many of the three digits are in the right place, and how many appear in the secret number, but not in the same place as guessed. Sounds simple, but complex logic is needed to figure out the secret number with as few guesses as possible.

The CTW disks are marketed by Apple and are available from Apple dealers. Each disk costs $50. The Mix and Match disk programs are in Applesoft. The programs on the other disks are in Integer BASIC, so they require either an Apple computer with Integer BASIC, or one with 64K (in which case Integer BASIC will automatically load into the extra memory when the disk is booted). Some of the programs also require game paddles.

The CTW programs, *Facemaker, My First Alphabet*, and those I reviewed in last May's column provide an excellent and varied set of software for introducing young children to computers. ©

trolled, military data processing technicians can write arbitrarily baroque programs that will cause it to do pretty much unpredictable things. Even if an enemy had access to the programs that guided the Turtle Task Team™, it is quite likely that they would find them impossible to understand, especially if they were written in ADA. In addition, with judicious use of the Turtle's touch sensors, one could, theoretically, program a large group of Turtles to simulate Brownian motion. The enemy would hardly attempt to predict the paths of some 10,000 Turtles bumping into each other more or less randomly on their way to performing their mission. Furthermore, we believe that the spectacle would have a demoralizing effect on enemy ground troops.

And what about munitions?

The Terrapin Turtle™ does not currently incorporate any munitions, but even civilian versions have downward-defense capability. The Turtle can be programmed to attempt to run over enemy forces on recognizing them, and by raising and lowering its pens at about 10 cycles per second, puncture them to death.

Turtles can be easily programmed to push objects in a preferred direction. Given this capability, one can easily envision a turtle discreetly nudging a hand grenade into an enemy camp, and then accelerating quickly away.

But what does it cost to install one?

The Terrapin Turtle is designed for installation at no cost by children and elementary school teachers. We feel that the military installation cost should be under $10,000 per unit.

I can think of no greater deterrent to all-out war than masses of robot turtles landing on the beaches and steadily moving towards the enemy.

Think of the tremendous opportunities for new patriotic songs: "When 4XQ7 Comes Crawling Home Again (Ta Raa, Ta Raa)," "Over There (Forward 20, Right 30), Over There," "How Are You Going to Keep Them Running POLY After They've Seen Paree?"

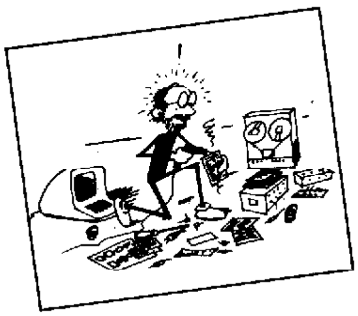
Instead of Basic training, the turtles will, no doubt, have to go through Logo training with procedures such as:

```
TO HUP :NUM1 :NUM2 :NUM3
REPEAT 4 [FD 10 WAIT 20]
HUP 2 3 4
END
```

Hats off to the Terrapin Patriots! May this be an ever safer world for turtles. ©

A Computer Language For Kids

Fred D'Ignazio, Associate Editor



In this column we have explored ways to make computers more accessible to kids. In the August to December columns, for example, we developed a computer "friend" for kids.

When the friend program is run, the friend's face appears on the TV screen. At first, the friend is asleep. "Ding! Dong!" goes a bell. The friend wakes up and winks. "I'm Ged," he announces. "Who's out there?"

The child answers by typing in her name. The friend greets the child and asks if she'd like to play a game. If she would, the friend gives her a menu of the games in its repertoire.

If the child is using a disk-based system, the friend starts the game automatically. If the child is using a cassette-based system, the friend helps the child load the game program from tape.

After the game, the friend comes back on the TV screen. "I hope you had fun," it says. It offers to play a new game with the child.

The Friendly Operating System

The friend is like a simple operating system. It is the interface, the middleman, between your child and the computer. It is a first attempt at making computers warmer, more human and personable.

In coming months, we'll be gradually expanding the friend's capabilities. Next month, for example, I will write about a way for the friend to learn more about the child. In a preliminary program the friend and the child will be "introduced." The child will give the friend personal information: name, age, the color of hair and eyes, address, phone number, likes and dislikes.

The friend will ask what kind of friend the child would like. The child will get a chance to mold the friend – to select the friend's name, shape, history, likes, and dislikes. If the child wishes, the friend can remain a computer. Or else the friend can become something completely imaginary and make-believe.

In fact, the child will be able to use the "Introduction" program to create several friends. The friends will have different characteristics and names.

If the child wishes, she can introduce the friends to each other.

Friendly Programming

The computer friends should liven up your child's computing. But they won't help with *programming.*

No matter how many games a friend has up its "sleeve," the child is never actually programming the computer. He or she is interacting with the friend and its programs. But not programming. Instead, in a way, the friend is programming the child.

This is one of the major drawbacks of the computer friends. They don't encourage children to write programs on their own. At least half of the value of the computer is unleashed when you program it yourself. Without that opportunity, your child is missing out on a lot.

Right now the friend is a friendly operating system. What we need is a friend that can also act as a *friendly computer language.* Then the friend can encourage the child to create, save, and run programs.

Beyond Logo

"Wait a second!" you say. "What about BASIC, PILOT, and Logo? These languages are easy to learn. They are friendly. They are perfect for kids."

Fred D'Ignazio is a computer enthusiast and author of several books on computers for young people. He is presently working on two major projects: he is writing a series of books on how to create graphics-and-sound adventure games. He is also working on a computer mystery-and-adventure series for young people.

As the father of two young children, Fred has become concerned with introducing the computer to children as a wonderful tool rather than as a forbidding electronic device. His column appears monthly in **COMPUTE!**.

My answer to that is: Do you have kids of your own? Do you teach kids? Have you ever tried to teach little kids how to use BASIC? Or PILOT? Or Logo?

I have two kids. My daughter Catie is almost seven. She's a first grader. My son Eric is three and a half. Eric spends his mornings at "Miss Eleven's Castle" (Evelyn's Day Care).

Both kids are whizzes at using the family computers. They have their own disks and tapes. They can turn the computers on and off, boot up disks, run programs, and key in the letters, numbers, and words the programs request. Both kids know all the special-function keys on the computer keyboard.

But try getting them to program? Forget it.

I can understand Eric's reluctance to program. After all, the kid doesn't even know how to read or write. If he gets 6's and 9's mixed up, and M's and W's, how can I expect him to master FOR/NEXT loops, string variables, subroutines, and arrays?

But Catie is a different matter. She reads Nancy Drew mysteries and "Choose Your Own Adventure" books. She is good at arithmetic, and she loves logic games, puzzles, and mazes. But she has no interest in programming.

Maybe it's just getting over the first hurdle. Unfortunately, Catie and I have been stuck on that hurdle for over two years.

The first hurdle is the first line of code in a program.

That first line is invariably a FOR/NEXT loop. The FOR/NEXT loop might do different things. It might print the message "CATIE LOVES MOWIE" a thousand times, all over the TV screen. (Mowie is Catie's kitty.) It might make the sound of a police siren or a dropping bomb, or the noise of water, or of crashing dishes. Or it might draw a drunken fly wandering across the screen.

What is Catie's reaction to all this? It's not positive, I'll tell you that.

Even if I get the fly to change into 16 different colors, Catie couldn't care less. After the first line of code, her reaction is sudden and dramatic. She gets hungry. Or she has to go to the bathroom. Or she has a headache. Or her spine dissolves and I get to watch her slide out of her seat and collapse into a puddle on the floor.

Or else she begins giggling and acts silly. She begins typing on the computer with her nose. Or her tongue.

This is an embarrassing situation.

On all sides we hear about friendly computers, computer literacy for kids, teaching kids to speak "computer" along with English. And here I am, a computer expert, a writer, an advocate for teaching computing to kids. So what do I do? I try to drag my kids into the computer age.

But they don't want to go.

Computer Literacy For Whom?

It's not so much that my kids resist me actively. It's just that they don't see the point. They have too much itching powder in their pants to make them sit still long enough to program.

At least using the languages available now.

But what if we created programming languages that incorporated the same ingredients as the best software designed for children? What are these ingredients? Quick response, for one. Other ingredients include: action, sound effects, pictures, colors. Quick mastery, a sense of power and control. Progress. Encouragement. Humor. These are qualities found in all good software for kids. But these qualities are not evident in programming languages. Even in PILOT. Even in Logo.

What Do You Think?

I hope I have lit some fires. Or started some fights.

What do you think? What kind of experiences have you had with your younger kids? Have they been similar to my experiences, or different?

Over the next few months as I continue to develop the computer friend and to write about other subjects, I plan to design and develop some prototype programming languages for little kids. The languages will be written in BASIC (or PILOT or Logo). They will be simple and experimental, something you can type into your computer and try yourself.

Also, the languages should contain the same qualities that make good programs so popular with kids. Maybe the programming will be in terms of colors, or sounds. Maybe in terms of shapes.

However it's done, the kids should be able to create programs themselves. They should be able to save, retrieve, and run those programs. The programs should not be trivial. They should do something. (Of course, they are doing *some*thing if they are teaching a child how to program.)

Most of all, the programming language should be fun for the kids to use. It should teach the kids that programming isn't something ugly that you have to do to get something nice. It's fun in itself. It's a way to express yourself, like coloring or playing music, or dancing.

The language shouldn't deter kids. It should encourage them to sit down and write a whole program. Even a short program.

Please write to me and tell me what you think. Send your letters to:

Fred D'Ignazio
c/o **COMPUTE!** *Magazine*
P.O. Box 5406
Greensboro, NC 27403

The Joy Of Joysticks

Atari/Commodore Game Controller Roundup

Tom R. Halfhill, Features Editor
Charles Brannon, Editorial Assistant

These joysticks and joystick substitutes will work with the Atari 400/800/1200XL computers, Commodore 64, VIC-20, and P Series computers, the Commodore Max Machine, the Atari VCS and 5200 video game machines, and the Sears Video Arcade. Some of the products, with proper adapters, also work with Apple, TRS-80, or Texas Instruments computers, and the Colecovision game machine.

Maybe you're playing a *Pac-Man*-type game....

Fleeing desperately from a relentless ghost, you make a break for the last energy pellet that will allow you to turn the tables on your pursuer. You try to round a corner in the maze, but suddenly find yourself slapping up against a wall. Why can't you turn? Blast that sticky joystick anyway! You've had it.

Or maybe you're playing a *Missile Command*-type game....

MIRVs and ICBMs are raining down and that infernal smart bomb is making straight for your last city. Quickly positioning the crosshair with the joystick, you take aim and fire your last ABM...and miss. If only you had a trackball like the one you're accustomed to in the arcades!

Or maybe you're playing an *Asteroids*-type game....

Hopelessly surrounded by an oncoming hailstorm of space debris, you yank back on the joystick to flip your spaceship into hyperspace, and find yourself dizzily spinning instead. Oh, for a hyperspace button like the one in the arcades!

The Joy Of Joysticks

Don't give up the spaceship – there is relief. A growing national obsession with home computer/video games has spawned an expanding market in custom game controllers. Only a year ago there were few alternatives to the common Atari-type joysticks supplied by the various manufacturers which use the Atari joystick standard. Now there are more than a dozen to choose from. The controllers covered in this overview were gathered after visiting computer stores, scanning magazine advertisements, and scouting new products at trade shows. While there are sure to be even more by the time this article appears, we tried not to leave any of the existing products out.

At first, it might seem that all joysticks must be more or less alike. Can there really be that much difference? After all, what is there to a joystick?

Externally, as the photos show, there is a wide range of configurations for joysticks (the name *joystick*, incidentally, originates from an early aviators' term for an airplane's control stick). Some joysticks are made to be hand-held and manipulated with a finger or two. Others are designed to rest on a tabletop and to be controlled with one hand. Some have hand-sized grips instead of short sticks. Some mount the fire button on the base, others on the stick, and still others have both.

Internally, there can be even greater differences. Some are constructed largely of plastic, others of metal. The construction largely accounts for a joystick's "feel." Since feel is a highly subjective reaction, we will avoid value judgments as much as possible. There is no substitute for trying a joystick yourself.

Some controllers, of course, are not joysticks at all. The push-button boxes are intended largely for *Asteroids*-type games, duplicating the arcade controls. Trackballs are at their best in games requiring rapid 360-degree movement, such as *Missile Command* and *Centipedes*.

And finally, a word about the standard Atari joystick. It's received some bad press, not all of it deserved. It's accused of being too fragile, unresponsive, and even ugly. The joysticks do wear out after months of heavy use, but this isn't all the joystick's fault. First, in our experience, many "broken" joysticks are really the victims of faulty cords. The cords are subjected to a lot of twisting and pulling, and the thin wires tend to fray and snap. A dead joystick can often be revived by replacing the cord. Keep this in mind when admiring a custom joystick's hefty construction: the standard cord is probably its weakest link.

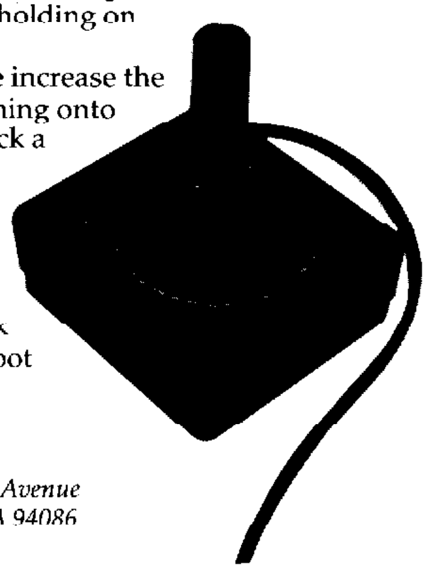
It might seem that all joysticks must be more or less alike. Can there really be that much difference?

Second, when an Atari joystick's joint or switches do break, it is often the fault of excessive flexing. Contrary to some beliefs, the Atari joystick *is* pretty responsive. Only a slight deflection is required to activate its switches. But its inherent stiffness, and the lack of any *tactile feedback* – that is, a positive click or snap when a switch makes contact – encourages people to wrench it harder than they have to. Games with slow joystick response, especially those written in BASIC, aggravate this problem.

Atari Joystick

Since the Atari joystick is the standard against which the others are most often compared, we'll start by pointing out that it's a two-handed instrument. Note that some joysticks permit one-handed operation, freeing the other hand for the keyboard (or for holding on to a chair).

Some people increase the leverage by jamming onto the end of the stick a PVC plastic "T" connector (available at hardware stores) or even a wine bottle cap. The Atari joystick includes a four-foot cord.

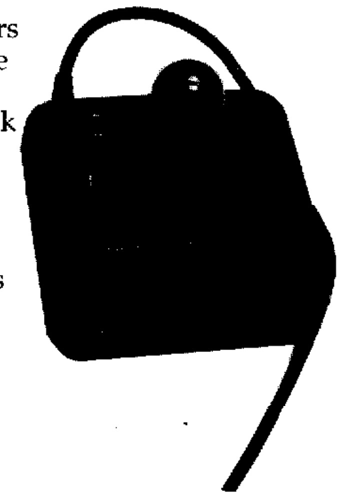


Atari Joystick
Atari, Inc.
1196 Borregas Avenue
Sunnyvale, CA 94086
$9.95 Each

Slik Stik

The Slik Stik is one of two joysticks by Suncom. Both resemble the Atari joystick, but incorporate some important differences. The Slik Stik's stick is only about half the height of the Atari's, but is topped by a jawbreaker-sized red ball for easy handling. And while the Slik Stik doesn't flex any more than the Atari stick, the action is more positive and you can feel a slight detent, or click. The fire button is very small but responsive.

The Slik Stik has a long six-foot cord reinforced at both ends with tough plastic collars where the cord joins the joystick base and plug. Suncom markets the Slik Stik as a direct replacement for the Atari joystick, and it is the only controller we reviewed which costs the same as the Atari product.



Slik Stik
Suncom, Inc.
270 Holbrook Drive
Wheeling, IL 60090
$9.95

Starfighter

Suncom's Starfighter, advertised as "The Ultimate Joystick," is very similar to the company's Slik Stik. However, Suncom claims it is more ruggedly constructed than their less expensive product, and it is guaranteed for two years instead of 90 days.

Where the Slik Stik has a ball-tipped controller, the Starfighter has a smooth plastic cylinder with a rounded top. It is taller than the Slik Stik, but still shorter than the Atari stick. The action is more positive, and the contacts in all eight positions can be distinctly felt. What's more, there are definite "stops" to the stick's movements, so it can't be damaged by over-twisting as the Atari joystick can. The Starfighter has the same convenient six-foot cord and reinforced connections as the Slik Stik.

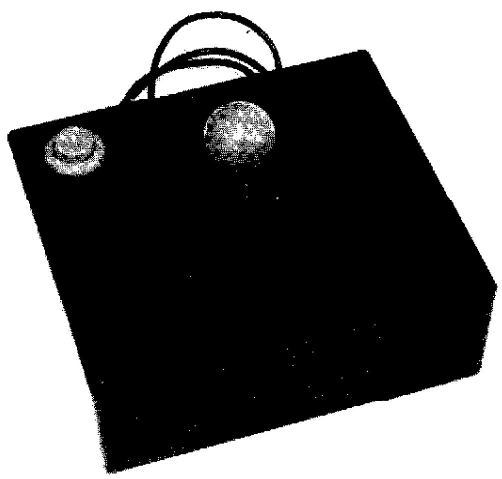
Starfighter
Suncom, Inc.
$16.95

Baylis Big Stick

The Baylis Big Stick is the largest controller we tested. Actually, its name is something of a misnomer; the stick itself is only two and a half inches high, including the large red ball on the tip. It is the base that is big – nearly eight inches square. Obviously, the Baylis is designed to be rested on a tabletop or lap and operated with one hand.

The base is heavy enough to permit this kind of operation, although it does tend to rock around a bit during heavy action. However, there seems to be plenty of empty room inside the base to add weights, if you want to customize it. The stick itself is a rigid steel shaft built to tough arcade standards.

The response is very flexible and positive, with more "travel" than many joysticks. The fire button also is a large, arcade-style device. The cord is on the short side, only two and a half feet



long, but since this oversized controller is not meant to be hand-held, this probably will not be a handicap.

The first Baylis Big Stick we sampled did not function in five of the eight directions. The internal switches were working perfectly, so the problem was traced to the cord. This is a perfect example of how even the most solidly constructed joystick can be paralyzed by the weakest link of any controller – its cord.

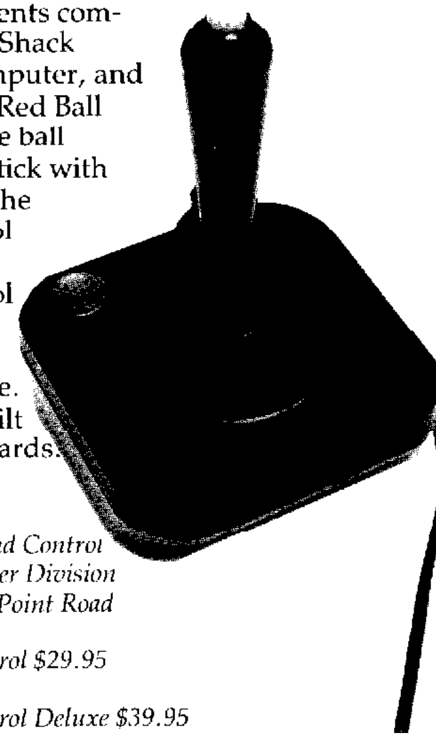
Baylis Big Stick
Released By:
Torrey Engberg Smith Co.
P.O. Box 1075
Glendale, CA 91209
$59.95

WICO Command Control

WICO's Command Control joystick is ruggedly built to arcade standards, with a steel shaft inside the plastic stick and metal parts at critical joints. This construction is not surprising, since WICO happens to be a major supplier of controllers for commercial arcade machines.

The Command Control joystick has a long "baseball-bat" handle, long enough to wrap your whole hand around. The action is smooth and flexible, with almost as much travel as the Big Stick. There is a small fire button on the tip of the stick and a larger one in the usual position on the base. A slide switch on the base selects between the two. The cord is five and a half feet long, strengthened with a plastic collar at the base end only.

WICO's product line includes two other joysticks, a trackball, extension cords, and adapters for Texas Instruments computers, the Radio Shack TRS-80 Color Computer, and the Apple II. The Red Ball joystick has a large ball mounted on the stick with the same base as the Command Control joystick, and the Command Control Deluxe features a batlike handle on a much larger base. All models are built to the same standards.



WICO Command Control
WICO-Consumer Division
6400 W. Gross Point Road
Niles, IL 60648
Command Control $29.95
Red Ball $34.95
Command Control Deluxe $39.95

Pointmaster

The Pointmaster is from Discwasher, a company whose best-known product is a popular cleaning system for phonograph records. The Pointmaster consists of a long plastic handle with a molded grip, attached with a ball joint to a plastic base. Since this unit is too light to use as a one-handed model, check to see if it is comfortable to use as a hand-held model, given its large size.

The stick is flexible enough, but there are no obvious contact points or "stops," so players should be careful not fo force the handle too far in the heat of video combat. Due to the stick's leverage and flexibility, precise positioning is sometimes difficult. The contoured fire button,

mounted on the tip of the handle, has almost no travel. When first toying with the Pointmaster, without plugging it in, we feared the button would have a "dead" feel. But actually, it turned out to be very sensitive and fast.

The cord is five feet long, reinforced with a collar at the base end only

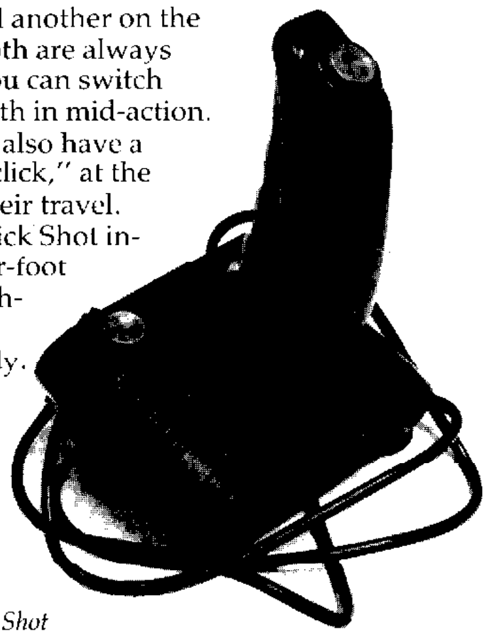
Pointmaster
Discwasher, Inc.
1407 N. Providence Road
Columbia, MO 65201
$16.95

Quick Shot

Spectravision's Quick Shot joystick has one unique feature that interested us immediately – the four rubber pads that are standard on other joysticks can be removed and replaced with four suction cups. This allows Spectravision to make the joystick small and light enough to be hand-held, yet still capable of being anchored firmly to a tabletop for one-handed use without resorting to a huge base or extra weights. We found, however, that the tabletop must be very smooth for the suction cups to stick, even if they are moistened.

Plastic construction dominates in the Quick Shot. The stick is a large, molded pistol grip that fits an adult's hand better than most of the other joysticks we tested. The action is flexible, with definite stops, although the contact points are hard to feel. There are two fire buttons, one on the stick and another on the base, and both are always "live," so you can switch back and forth in mid-action. The buttons also have a detent, or "click," at the bottom of their travel.

The Quick Shot includes a four-foot cord strengthened at the base end only.



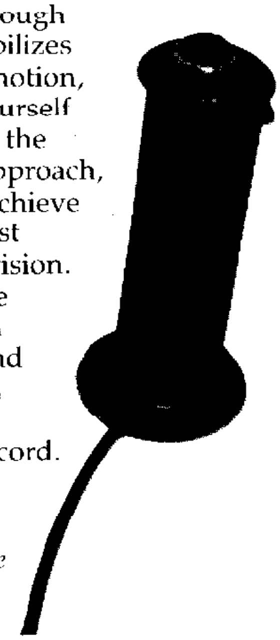
Quick Shot
Spectravision
39 W. 37th Street
New York, NY 10018
$14.88

Le Stick

Le Stick is the most unusual joystick we tested. Datasoft claimed in early magazine ads that Le Stick was adapted from Air Force designs for advanced controllers. Le Stick consists only of a joystick – no base. Constructed of a pliant, rubber-like plastic, the handle incorporates four mercury switches which are activated by tilting. That is, tilting the handle forward causes the screen object to move up, tilting it backward moves the object down, and so forth.

This ingenious approach seems to have several advantages: without a mechanical connection to a base, flexibility is unlimited; there is no ball joint to wear out; true one-handed operation is possible, since the fire button is tip-mounted; and the joystick is very light.

However, since the joystick has no "self-centering" or definite "up" position relative to an attached base, it can be difficult to maneuver for those accustomed to conventional joysticks. For example, our untrained hands found it difficult to tilt horizontally without mixing in some vertical motion, and vice versa. Although squeezing the handle immobilizes the sensor and cancels any motion, it can be hard to re-orient yourself without taking your eyes off the screen. As with any novel approach, practice will be required to achieve mastery – we suggest you test Le Stick before making a decision. Our last suggestion – beware the "grip of death" when, in panic, your hand clinches and immobilizes the joystick ... a calming challenge.



Le Stick has a four-foot cord.

Le Stick
Datasoft, Inc.
19519 Business Center Drive
Northridge, CA 91324
$39.95

Starplex Video Game Controller

Unlike the joysticks reviewed, the Starplex Controller and the KY Enterprises box covered below are not really general-purpose devices suitable for all types of computer games. Instead, the Controller is intended largely for one game – *Asteroids*. The button layout is designed to simulate the controls on the commercial arcade version. Thus, we find buttons labeled "Left," "Right," "Up," "Down" (Hyperspace), and "Fire." These correspond to the rotational, rocket, and panic buttons on the arcade machine.

The Starplex Controller fulfills its task very well. Anyone accustomed to playing *Asteroids* in the arcades will feel much more at home with these large, sensitive buttons than with a joystick. One interesting feature is the "Astroblast." Selecting this option with a slide switch allows automatic repeat when the fire button is held down. In other

Over thirty years of down-to-earth experience as a precision parts manufacturer has enabled Star to produce the Gemini series of dot matrix printers—a stellar combination of printer quality, flexibility, and reliability. And for a list price of nearly 25% less than the best selling competitor.

The Gemini 10 has a 10″ carriage and the Gemini 15 a 15½″ carriage. Plus, the Gemini 15 has the added capability of a bottom paper feed. In both models, Gemini quality means a print speed of 100 cps, high-resolution bit image and block graphics, and extra fast forms feed.

Gemini's flexibility is embodied in its diverse specialized printing capabilities such as super/sub script, underlining, backspacing, double strike mode and emphasized print mode. Another extraordinary standard feature is a 2.3K buffer. An additional 4K is optional. That's twice the memory of leading, comparable printers. And Gemini is compatible with most software packages that support the leading printers.

Gemini reliability is more than just a promise. It's as concrete as a 180 day warranty (90 days for ribbon and print head), a mean time between failure rate of 5 million lines, a print head life of over 100 million characters, and a 100% duty cycle that allows the Gemini to print continuously. Plus, prompt, nationwide service is readily available.

So if you're looking for an incredibly high-quality, low-cost printer that's out of this world, look to the manufacturer with its feet on the ground—Star and the Gemini 10, Gemini 15 dot matrix printers.

star micronics • inc

MAKING A NAME FOR OURSELVES

1120 Empire Central Place, Suite 216, Dallas, TX 75247
For more information, please call Bob Hazzard, Vice President, at (214) 631-8560.

words, now you can machine-gun the nasty asteroids. This feature requires an AA battery to be installed inside the controller.

The Starplex also works well with *Space Invaders* and other games requiring simple up-down or left-right movement. For games that demand complex 360-degree movement, stick with a joystick. Obviously, you'll have to decide if you can use this type of controller often enough to justify its cost.

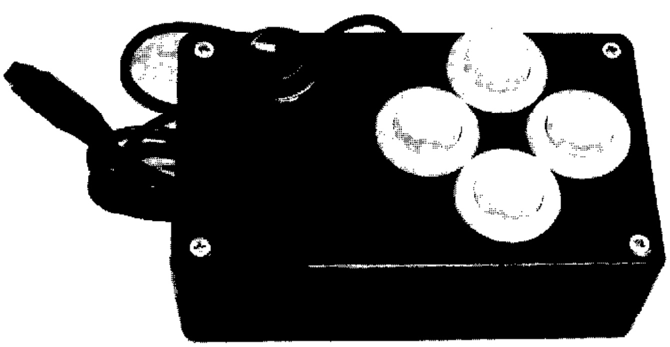
The Starplex is light enough to rest on a lap, and stable enough to hold still on a tabletop. It has a four-foot cord reinforced at the base end only.

Starplex Video Game Controller
Starplex Electronics, Inc.
E23301
Liberty Lake, WA 99019
$29.95

Fingertip Controller

This controller is very well constructed, with a heavy metal box and five large, springy, arcade-style buttons. The buttons are unlabeled, but the white ones correspond to up, down, left, and right, while the red one is the fire button.

Although you can achieve diagonal movement by simultaneously pressing both a vertical and horizontal button, the Fingertip Controller seems most suited to games with simple up-down or left-right movement, such as *Space Invaders*. Like the Starplex Controller, it also works well for *Asteroids*, but with a quirk – it's left-handed. That is, your right hand controls the rotational movement while your left hand hits the fire button,



just the opposite of the arcades.

As per the instructions, it's easy to adjust the sensitivity of the buttons by opening the box and bending the spring switches. The Fingertip Controller has a five and a half-foot cord.

Fingertip Controller
KY Enterprises
3039 East Second Street
Long Beach, CA 90803
$26.95

Command Control Trackball

True arcade fans have been hungering for one of these for a couple of years now. Commercial arcade games which use trackballs – such as *Missile Command* and *Centipedes* – work okay when trans-

lated to joysticks in home versions, but the "feel" just isn't there. And since the avid arcade fan strives to re-create the arcade experience as closely as possible, joysticks sometimes just don't quite measure up.

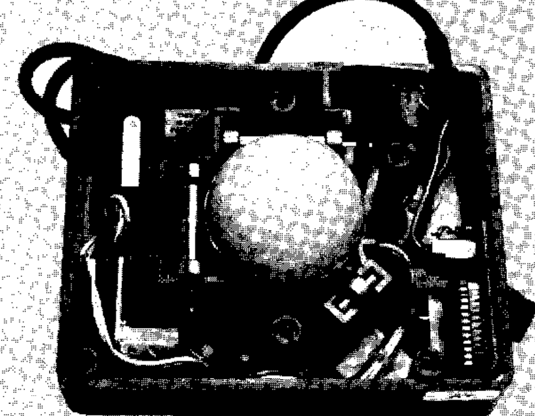
Since WICO supplies trackballs for commercial arcade machines, you would expect the company's home version to be similarly well-constructed – and you won't be disappointed. The heavy billiard-style ball rotates quite smoothly and "coasts" with a good spin. This is due to high-quality steel shafts with ball bearings (see the accompanying sidebar and inside photo describing how the trackball works). Even the five-foot cord is extra heavy-duty. The trackball's inherent weight and rubber footpads keep it from sliding around on a tabletop, and the fire button is the same as those found on WICO's joysticks.

As an example of what a trackball can do in a game demanding fast 360-degree movement, one of our testers tried it out on Atari's *Missile Command*. His former high score was 39,000. With the trackball, after a few warm-up games, he scored 66,000.

Command Control Trackball
WICO
$69.95

WICO Trackball: The Inside Story

Ottis Cowper, Technical Editor



Most joysticks operate by opening or closing switches as the handle is moved. In the standard Atari configuration, four switches provide a four-bit binary number for control of motion in eight distinct directions.

Exceptions to this rule are joysticks such as those used with the TRS-80 Color Computer which use a pair of potentiometers (variable resistors) to provide varying voltages which must be converted by the computer to meaningful binary values. Such joysticks are essentially two-dimensional game paddles. The WICO trackball uses an altogether different technique. Let's take a look inside this rather unconventional game controller to see how it works.

The ball, which is remarkably similar to a billiards cue ball, rests on three rollers with ball bearings for smooth motion. The two larger rollers, one placed vertically and one horizontally, both have a shaft with a slotted disk on one end. These disks pass through the gap in an electronic device known as a *photon-coupled interruptor* and herein lies the key to the trackball's operation. A photo interruptor consists of a light-emitting diode (LED) and a phototransistor separated by a gap. As long as the gap is not obstructed, light from the LED strikes the phototransistor and turns it on. If the light is blocked, the transistor turns off.

As the slotted disk rotates, an alternating series of solid sections and holes passes through the gap, causing the transistor to toggle on and off as light from the LED is alternately blocked and allowed to pass. (The photo interruptors make it possible to determine in which direction the disk is rotating.) Since the transistor can be thought of as an electronic switch, this has the same effect as pushing the joystick handle in one direction, except that the input is much faster and smoother.

For games which require rapid motion all over the screen, the trackball is a major improvement, although the standard joystick is probably more suitable for applications which require precise positioning.

TG Trackball

This trackball should be on the market by the time you're reading this issue. The unit we tested was a prototype that we obtained at the COMDEX trade show in Las Vegas. TG Products also is introducing an Atari plug-compatible joystick, but we were unable to obtain one of these for testing.

The TG Trackball works much like the WICO Trackball, using LEDs and phototransistors to detect the ball's spin. The plastic ball glides less smoothly than the WICO's, however, and has much less tendency to coast. Approximately one third of our testers preferred this "feel" for fine positioning, so this is purely a personal matter that should be tested by the purchaser. Inside, the TG Trackball supports the "billiard ball" on plastic shafts without ball bearings. It might be a good idea to lubricate these shafts to reduce excessive wear if this hasn't been done in production models.

The trackball's extra-heavy cord is just short of five feet and is reinforced at both ends.

TG Trackball
TG Products
1104 Summit Avenue
Suite 110
Plano, TX 75074
$64.95

PROGRAMMING THE TI

C. Regena

Write Your Own Games

Some tips on getting the most out of your TI when writing games.

You have probably discovered that one of the fun things to do with your TI-99/4A is to play games. In fact, many people who wanted one of the popular game machines have discovered that for about the same amount of money they could have a *computer* and still be able to play games. Many of the games written for the TI-99/4A are arcade quality – that is, they have good graphics and fast action.

The programs on the command modules can be programmed in UCSD Pascal, TMS9900 Assembly, and Graphics Programming Language (GPL). These languages take maximum advantage of the color, graphics, sound, and speech capabilities of the computer. GPL is an excellent language for drawing graphics and allows the speed of an assembly language.

To program your own games with fast, smoothly-moving objects, you will want to use TI Extended BASIC. It allows you to use up to 28 "sprites." You may define the shapes of the sprites and designate a certain magnification. You may also specify the sprites' speed. The row velocity and the column velocity may vary from -127 to +127, and by specifying numbers for both velocities you will get a diagonal movement. Sprites "wrap" at the edges of the screen, so you don't need to worry about "crashing" your program on edge conditions. With *one* CALL SPRITE statement you can define the sprite number, shape, color, position, and speed.

TI Console BASIC (the BASIC built in with no accessories or peripherals) is a language powerful enough that you can design a variety of fun games with it. If you have moving objects, however, they have to move a square at a time and thus will have jerky movement. Depending on the number of objects, BASIC games tend to be slow; however, I have seen several fast action games that really require nimble fingers.

Whether you are writing a game in TI BASIC or in TI Extended BASIC, I can offer a few programming tips. Keep in mind that the best way to learn is to actually start programming – and playing.

Randomness

Probably a central tool in computer games is the machine's ability to choose things randomly. Most computers have the command RND, but each computer has a slightly different syntax (way of writing the command). On the TI-99/4A, RND represents a random number between zero and one. Turn on your computer, press any key to begin, and press 1 for TI BASIC. Now type in PRINT RND and press ENTER. The computer will print a decimal fraction (to ten places). Usually in game situations you won't want a fraction, so multiply that fraction by a number. For example, multiply RND by 10 like this: PRINT 10*RND or PRINT RND*10. Now you will get ten times that decimal fraction.

You probably want just the whole number part of that mixed decimal number. Use the INTeger function to get the whole number. PRINT INT(10*RND). If you keep trying this command, you will get numbers from zero to nine. Remember, INT truncates the decimal portion; it does not round the number. Suppose you really wanted random numbers from one through ten. The command would be: PRINT INT(10*RND) + 1 or PRINT INT(10*RND + 1).

One more step. Assume you want a number N to be a random number between 10 and 20, inclusive. 20 - 10 = 10. There are 10 numbers plus 1 ("inclusive"). The command could be N = INT(11*RND) + 10. The portion INT(11*RND) will give you numbers from 0 to 10; then you add 10 to get numbers from 10 to 20.

Now try this short program:

```
100 FOR I=1 TO 10
110 PRINT INT(10*RND)+1
120 NEXT I
```

RUN the program. RUN it again. And again. The program is printing ten random numbers from 1 to 10. However, you'll notice that each time you run it, you get the same numbers in the same order. You need to add the line: 105 RANDOMIZE.

The RANDOMIZE command mixes up the numbers so that each time the program is run you will get different numbers – and that's what you want in a game. The *User's Reference Guide* indicates that the RANDOMIZE statement only needs to be somewhere in the program to generate different numbers; however, I have found that one RANDOMIZE statement at the beginning of a program does not always work. It is better to use the RANDOMIZE statement just before you use the statement containing RND. Note: If you are debugging a program, you may want to leave RANDOMIZE out so you'll know exactly what numbers your program is choosing. Debug your program, then add the statement and test it.

Moving Objects

In general, the fewer moving objects you have in your game, the faster the action can be, and the logic will be a lot less complex. Also, each moving object should be specified by only one character number so you don't have to use up valuable time by building an object out of several characters. To move an object in TI BASIC you need to erase the object in the first position (replace it with a space) and draw it again in the second position - each move takes two statements.

Player Input

There are two main ways the computer can understand what you want: by your using the joysticks or pressing keys on the keyboard. Your game may be designated for joysticks only, keyboard only, or both. Because of the logic involved, a game using both methods of input will be slightly slower in response; and depending on the branching sequence, one of the methods will be slower than the other.

Joysticks may be easier to use to learn a game, especially if the player is used to a video game using joysticks. My own children, and many other players I know, prefer using the keyboard for *TI Invaders* and *Munchman* because the joystick response is considerably slower than the keyboard response.

The keyboard action is easy to learn because there are standard arrow keys for all games designed for the TI-99/4A. Programmers writing games for other computers often choose their own favorite keys to use, and the directions are different for each game. On the TI-99/4A, the arrow keys are E (up), X (down), S (left), and D (right), with the shooting key either the ENTER key or the period key. If there are two players, the standard arrow keys on the right half of the keyboard are I, J, K, and M. The TI-99/4 owners have a slight advantage here – there is an overlay available for the old keyboard that shows the arrow keys, and it is easier to use the old keyboard for two-player games.

The TI joysticks (wired remote controllers) come with a little instruction book with some sample programs. The main command is CALL JOYST(K,X,Y), which returns an X and Y value for the position of the joystick, where X and Y may be 4, -4, or 0.

To detect keys pressed on the keyboard, use the CALL KEY command. This command is like the GET command in other BASIC languages. The form is CALL KEY(0,KEY,STATUS) where 0 means to scan the whole keyboard. STATUS is a variable name (it could be ST or S, or whatever you wish) which will return whether a key has been pressed or not. KEY is a variable name (again, use whatever you wish) that will return the ASCII code of the key pressed, such as 13 for the ENTER key, 65 for the letter A, 69 for the letter E, etc.

By using IF statements, you can check which key was pressed and branch accordingly. You can also GOTO the CALL KEY statement for other keys to make the computer act as if it is ignoring all responses except the keys allowed. Here is a sample using arrow keys:

```
100 CALL KEY(0,K,S)
110 IF K=69  THEN 1000      (up arrow)
120 IF K=68  THEN 2000      (right arrow)
130 IF K=88  THEN 3000      (down arrow)
140 IF K=183 THEN 4000      (left arrow)
        ELSE 100            (any other key will be ignored)
```

Remember, there are several ways to program the same procedure; this is just one way. You may prefer to use "not equal" signs or a split keyboard and an ON GOTO statement.

A split keyboard approach scans half the keyboard using CALL KEY(1,K1,S1) or CALL KEY(2,K2,S2). The key codes returned for up, right, down, and left are 5, 3, 0, and 2. A sample program using the split keyboard is:

```
100 CALL KEY(1,K,S)
110 IF (K<0)+(K>5)THEN 100
120 ON K+1 GOTO 3000,100,4000,2000,100,1000
```

Line 110 makes sure the K value is in the right range; the key value must be from 0 to 5. All other keys are ignored. Line 120 branches according to which key was pressed. The keys corresponding to 1 and 4 were not acceptable, so they return to the CALL KEY statement. If you want to try out either of these programs, add the following lines, then RUN and try pressing various keys.

```
1000 PRINT "UP"
1010 GOTO 100
2000 PRINT "RIGHT"
2010 GOTO 100
3000 PRINT "DOWN"
3010 GOTO 100
4000 PRINT "LEFT"
4010 GOTO 100
```

There is a slight problem in testing for zero on the TI-99/4A console. Use logic such as IF K+1<>1 rather than IF K<>0. Also, some of the split keyboard codes are different for the TI-99/4A than for the TI-99/4. It is better not to use the comma, period, semicolon, slash, space bar, ENTER, SHIFT, B, and C so that programs may be used on either console. ©

Shamus For Atari

Tom R. Halfhill, Features Editor

You're prowling along the corridors of yet another unexplored room, searching for the key to the Shadow's lair…

Suddenly you are attacked by a hunting pack of Whirling Drones, Robo Droids, and the especially deadly Snap-Jumpers. Frantically dodging their molecular disruptors, you hurl several of your contraband Ion-Shivs, blasting them to fragments. Now you're free to pick up the key they were guarding, and you hope that it fits the lock you encountered in that other room far behind you.

But you've dallied too long in this chamber. From out of nowhere descends the Shadow himself, protected by Tri-Gamma body armor impervious to your Ion-Shivs, and he's bent on revenge for the destruction of his henchmen. You break for the exit, but stumble into a wall instead…and instantly disintegrate.

A Blend Of Arcade And Adventure

That's a typical example of how Synapse Software's game *Shamus* is played – and a typical example of how it usually ends as well, since this game is extremely hard to beat. In fact, my guess is that it would take months of frequent play before any mere human could succeed in locating the Shadow's lair and destroying the elusive arch-enemy. This is a game for true addicts.

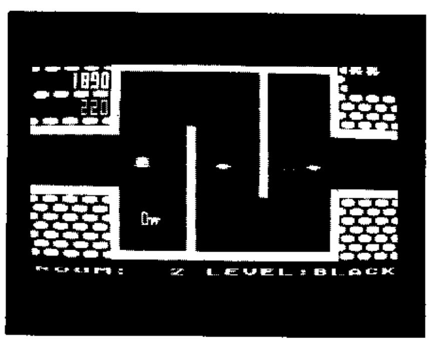
Shamus (pronounced "SHAW-muss" or "SHAY-muss," slang for *detective*) is a one-player game available on disk or cassette which requires at least 16K RAM and a joystick. Programmed by William Mataga, *Shamus* combines the puzzle-solving and exploration features of a graphics adventure game with the fast action of an arcade-style shoot-'em-up.

The object of the game is to locate the hidden lair of a creature known as the Shadow, and then to destroy him in a final struggle. Locating this lair is not easy. There are four levels of rooms to explore, and the only way to advance to the next level is to find the proper key for the proper lock. The locks and keys are color-coded and scattered throughout the rooms, forcing you to wander around, picking up keys and trying them on the various locks.

To give you some idea of the complexity of this task, each level contains no less than 32 rooms – according to the manual. Actually, in my aimless wanderings, I encountered rooms numbered as high as 37 on one level. This means there could be nearly 150 rooms!

The graphics and sound effects in *Shamus* are beautifully done. The game boots up from the disk or cassette with a very good rendition of the theme tune from the old *Alfred Hitchcock Presents* TV show. You then choose from four degrees of difficulty ranging from "novice" to "expert." The manual is absolutely correct when it states that each degree is significantly harder than the last. After briefly sampling the higher degrees, I stuck with "novice" and still found myself outmatched.

You start off in Room 0 on the first level. Your joystick controls a little man in a fedora (after all, what kind of detective would



Going for the key to the next level while dodging Whirling Drones in Shamus.

you be without a brimmed hat to pull down over your eyes?). Each of the 32 (or 37, or whatever) rooms on each level occupies a full TV screen. To move to another room, you simply head for a door and walk (or run, as is frequently the case) off the screen. Instantly, the next room appears.

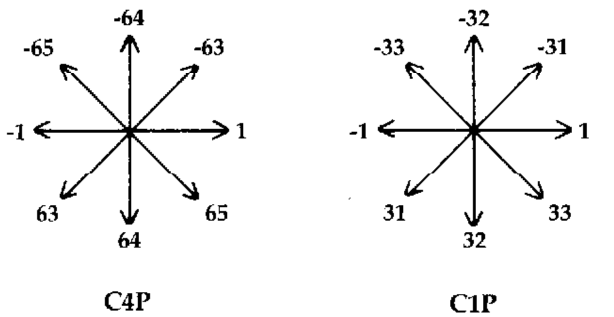
Shamus uses several redefined character sets, and the graphics are among the best I've seen on the Atari. Joystick response is instantaneous, and very often a half-dozen or more multicolored objects will be moving around at once.

These objects, by the way, are the Shadow's henchmen. Searching for keys and locks in scores of rooms spread over four levels would be hard enough, but these creatures are always there to make your life even more difficult. The easiest to dispose of are the Whirling Drones, little pinwheel-shaped machines that home in on your presence. The Robo-Droids aren't too bad either, although they're a headache when attacking in droves with the Whirling Drones. Far more dangerous are the Snap-Jumpers, shifty little critters who move in short leaps in the blink of an eye.

If any of these henchmen shoot you with their molecular disruptors, or even touch you, it's goodby. Your main defense is your inexhaustible supply of Ion-Shivs (Ionic-Short High Intensity Vaporizers). You can throw these in any direction by pressing the fire button while aiming the joystick, and they'll disintegrate anything. Another defense is dodging or even fleeing, but watch out – if you brush against a wall, you'll be instantly zapped to atoms.

By far the most dangerous obstacle, though, is the Shadow himself. If you stay in one room too long – say, half a minute or so – he appears out of nowhere and tries to destroy you with his deadly touch. Since the Shadow wears Tri-Gamma body armor, the statement POKE X,32. The number 32 is the ASCII code for a blank. You must change X to the new location of the car on the screen and then POKE X, 0 since 0 is the number for a car. When you change the car's position, you simply execute the statement X=X+Z, where the variable Z depends upon the direction you wish to move and whether you are using a C1P or a C4P. Here are the values:

Figure 2.



Thus, if we wanted to move down, the statement would be X=X+64 for a C4P and X=X+32 for a C1P. Program 3 uses these constants from their tables to move a cross in random directions. The program does not check for the edges of the screen to see if the cross has travelled past its boundaries, so if you don't press CTRL-C before it goes far, the program might hang up.

Program 1.

```
10 REM FILL SCREEN WITH AIRPLANE
20 P=PEEK(57088):REM LOOK AT KEYBOARD
30 ST=53315:EN=54205:REM C1P VALUES
40 IFP<129THENST=53376:EN=55295:POKE56832,1:R
   EM C4P VALUES
50 FORLO=STTOEN:POKELO,236:NEXT
60 GOTO 60
```

Program 2.

```
10 REM PUT SPECIFIED NUMBER OF MEN ON SCREEN
15 REM AND COUNT THEM. THERE IS AN INCREASING
    AMOUNT OF ERROR
17 REM AS THE NUMBER OF MEN IS INCREASED DUE
   TO THE FACT THAT
18 REM THE MEN ARE PUT IN THE SAME BOX AS ONE
    ANOTHER.
20 SU=0:ST=53315:EN=54205:X=24:Y=28
30 IFPEEK(57088)<129THENST=53376:EN=55295:X=6
   4:Y=30:POKE56832,1
40 INPUT"NUMBER OF MEN";ME:FORCO=1TO30:PRINT:
   NEXT
50 FORCO=1TOME:POKEST+INT((EN-ST)*RND(1)),240
   :NEXT
60 FORCO=STTOEN:IFPEEK(CO)=240THENSU=SU+1
70 NEXT:PRINT"THERE WERE";SU;"MEN ON THE SCRE
   EN."
```

Program 3.

```
10 REM MOVE CROSS
20 FORX=1TO8:READP(X):NEXT:LO=54016
30 IFPEEK(57088)>128THENFORX=1TO8:READP(X):NE
   XT:LO=53775
40 DATA1,65,64,63,-1,-65,-64,-63
50 DATA1,33,32,31,-1,-33,-32,-31
55 FORX=1TO30:PRINT:NEXT
60 POKELO,219:FORX=1TO30:NEXT:POKELO,32
70 LO=LO+P(INT(RND(1)*8+1)):GOTO60
```

COMPUTER CALCULATORS

Jim Butterfield, Associate Editor

Number conversion, masking, even translations of floating point variables are possible when you use the more sophisticated "programmer's calculators." Here are some techniques for using various types of calculators when your computer is doing other things.

Why have a calculator when you already have a computer? Indeed, why would you need a special calculator when the simple four-function units will do all the arithmetic you might need?

The answer is: convenience. It's sometimes handy to be able to zip through a quick calculation and get the results in binary, hexadecimal, octal, or whatever. If your computer isn't handy (or someone is playing space invaders on it at the moment), there are questions you can work through if you have a calculator to help.

But make no mistake about it: the sophisticated machines are not indispensable. You can do the job with no calculator at all. You can use a simple four-function unit. You can do useful calculations with a simple programmable unit, entering programs to do the work. Or you can get a "programmer's calculator."

No Calculator

Honest, there are still people out there who add and subtract – and even multiply and divide – without a calculator of any sort. There are programmers who know how to add and subtract in hexadecimal or octal. It's probably good for you to know number systems from firsthand experience.

For example, to convert a decimal number to hexadecimal, divide the number repeatedly by 16. The remainder from each division is a hexadecimal digit; you'll generate the digits from right (low order) to left. So 200 decimal is converted as follows: 200 divided by 16 gives 12 with a remainder of 8. Our last hex digit is 8. Continuing: 12 divided by 16 gives nothing with a remainder of 12. Our next hex digit is 12, which we write as C. The hex value: C8.

Going the other way – from hexadecimal to decimal – is just as easy. We take the digits from the left. After we pick a digit, we see if there are any more. If so, we multiply by 16 and add the value of the next digit. So hex C8 becomes 12x16 +8 or 200 decimal.

On The Computer

It's not hard to write a program to do the conversions. The problem is this: we usually have a program half-written on our machine at the moment we wish to convert something. Loading a program is out; we'd lose our work in progress. For this reason, we usually use direct statements.

From hex to decimal, we usually multiply by powers of 16. Thus, the hex address 027A is evaluated by the direct statement PRINT 0*4096 +2*256 +7*16 +10. 4096 is 16 to the third power; 256 is 16 squared.

From decimal to hex, there's no fixed method. Some people divide the number by 4096 to get the first digit. For example, 59468 divided by 4096 yields 14.5185547 – 14 is a letter E, our first digit. After that, there are a variety of methods: subtracting out the high amount (59468 - 14*4096) is one way, and using the fractional value (.5185547 x 16) is another. In either case, a little work starts to reveal the following digits.

The Four Function Calculator

Most calculators aren't very good at giving you remainders after a division. They will happily tell you that 59486 divided by 16 is 3716.75, rather than that it gives 3716 with a remainder of 12. For this reason, many users like to work decimal to hex conversions from the high-order end.

For a 16-bit number (0 to 65536), divide by 4096. Repeat four times: note the integer value, which is your hex digit; subtract that value to give a fraction; and multiply by 16.

So for 59468 we divide by 4096 to get 14.5185547. Subtract the 14 – that's E, our first digit – and multiply by 16 to get 8.296875. Subtract the 8 – now we have E8 as the start of our hex value – and multiply by 16 to get 4.75. Subtract the 4 – our number is almost complete at E84 – and multiply by 16 one last time. Our final digit will be close to 12, hex C, so we may write our final hexadecimal value as E84C.

Hexadecimal to decimal is much easier. Take the first digit's value. If there are any more digits, multiply by 16 and add the next digit. Keep going until you have the value. Hex E84C works quickly to its decimal value via these numbers: 14, 232, 3716 and finally 59468.

The Programmed Calculator

With a programmable unit, we can place the above calculations into a program and have the steps done automatically for us.

Many programmable units have a FRAC function which simplifies the sequence of steps. FRAC is the opposite of the INT function. For example, FRAC of 8.296875 yields .296875 and allows us to save a subtraction step in the conversion.

Since most programmable calculators can't input, calculate, and display hexadecimal digits, it is not possible to show (or enter) a value such as E84C. The usual way to overcome this problem is to use a "double digit" hex display, so that E84C will be displayed as 14080412 – the 14 standing for E, the 08 for 8, and so on.

The TI Programmer

The Texas Instruments Programmer is a special-purpose calculator which allows input and display of decimal, hexadecimal, and octal numbers, together with easy conversion between them. Simple four-function arithmetic can be performed, plus logical functions such as AND and OR.

The calculator is not programmable. It has a memory which allows storing a number or accumulating a total. In decimal mode, fractions can be entered – for example, 36.25 – but no fractions can be used in the other number bases.

Relative branch address calculations can be performed by simple subtraction. And the conversions are very simple – just push a button.

The Hewlett-Packard 16C

The H/P 16C is a more expensive calculator, but has many more features. Not only does it have all the logical functions (AND, OR, XOR, and NOT), but it also has an extensive set of Rotate and Shift commands, including a Carry flag. There are commands to set, clear, or test individual bits within a number, and functions which create a "mask" of any number of high bits or low bits.

Conversion of numbers is simple, of course. The 16C will copy with negative numbers, if you wish. You may set it to: unsigned numbers; twos-complement signed numbers (the "usual" way of holding signed numbers); and ones-complement signed numbers, a relatively rare way of representing negative values. We may limit the calculator to a specific number of bits, so that -1 will be shown as hex FFFF in 16 bits or FFFFFF in 24 bits.

The 16C has an "integer" side, with decimal, hexadecimal, octal, and binary display modes; and a "floating" side, which allows decimal numbers complete with fractional parts. The floating mode is good for conventional calculation, although it has no scientific functions.

A remarkable thing about this calculator is that it allows you to convert between floating point numbers and floating binary notation. This is a good trick, since it involves generating an exponent and a mantissa. Not everyone needs this feature, but it's surprising to see such a powerful calculation available.

Floating Point To Decimal

Let's work through this calculation on a variable in Microsoft BASIC. Somewhere in BASIC is a floating-point value stored as hex 81 49 0F DA A2. The 81 is the exponent, and the rest is the mantissa. Let's find its decimal value. Press "f" "2's" to ensure that the machine is in twos-complement signed mode; press "HEX" 30 "f" "WSIZE" to put us into the hexadecimal mode with enough bits to work on.

Now we enter the mantissa: 490FDAA2. Microsoft drops the high bit from positive numbers; we must put it back by pressing "ENTER" 1F "f" "sb" (for set bit 1F, or bit 31). Now we should see C90FDAA2, our corrected mantissa. Now for the exponent: type in the 81. To adjust for differences between Microsoft and Hewlett-Packard, we must subtract hex A0: type "ENTER" A0 "-" (minus).

The display will show something like FFFFFFE1, our adjusted exponent. We're all ready: press "f" -"FLOAT" -8 and we'll see the value: 1.57079633, or one-half pi. We can go the other way just as easily: press "HEX"; adjust the exponent by adding A0; flip to the mantissa by pressing "X-Y"; knock out the high bit by typing 1F "f" "CB". Easy. Remarkable.

The 16C is programmable. The above sequence of operations, or any other, may be entered as programmed instructions so that a simple key sequence (for example, GSB A for GOSUB A) will trigger the whole computation. The calculator has continuous memory; even if it's switched off, the program – and for that matter the data – will remain.

The calculator has many memory locations. How many? That depends on two factors. First, the size of the programs you have stored, if any. Each program instruction takes away from memory space. Second, the "word size" that we have selected. If we decide to work only with eight-bit numbers, for example, we'll have a very large number of memory locations – up to 203. With the maximum size number – 56 bits – or floating

New Marketing For CompuServe

Computer retailers, software stores and major retail chains can now sell the CompuServe Information Service (CIS) through a new retail marketing program.

CIS can be used by all personal computers and by many computer terminals.

The CIS "starter kit" contains a three-ring vinyl binder, a user ID number and password for five free connect time hours on CIS, an easy to read *User's Guide*, and an introductory subscription to *Today* magazine. The starter kit also contains local phone numbers for connecting a personal computer to CIS, access instructions, a dictionary of computer terms, commonly asked questions and answers about CIS and rate information.

The starter kit has a suggested retail price of $39.95.

CIS offers shopping and banking at home, electronic mail and realtime communications, current and historical stock market and commodities information, family information and education, up-to-the-minute news and weather, electronic games, bulletin boards, private user groups and computing power for programming activities.

CompuServe Incorporated
5000 Arlington Centre Boulevard
Columbus, OH 43220
(614)457-8600

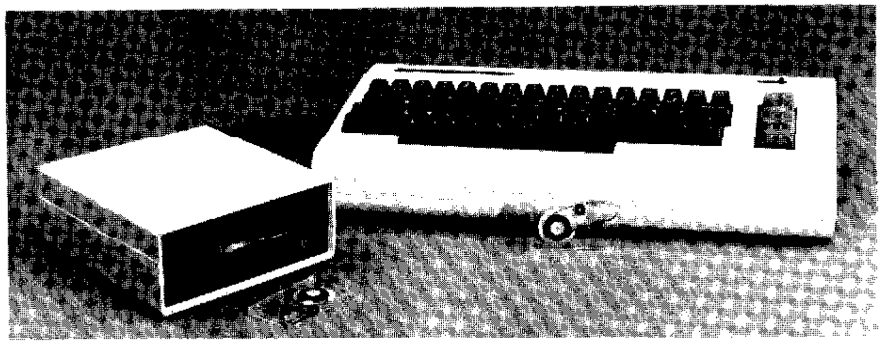
Mass Storage Peripheral For Commodore 64 And VIC-20

Exatron has announced its Commodore 64 and VIC-20 compatible Stringy Floppy mass storage peripheral. The new system, named ESF-20/64, consists of two units: a miniature endless-loop tape cartridge and a precise electronically direct-drive transport mechanism which plugs into the serial bus connector on the Commodore 64 and VIC-20.

Major features include reliability, speed, compact size, and easy interfacing with no changes to the hardware or software of either computer. The commands used to operate the ESF-20/64 are incorporated into the Commodore 64 and VIC-20 and are fully explained in the *VIC-20 Computer Guide*.

The tape transport operates at a speed of five inches per second and has memory capacity of up to 64K bytes. Previously stored data is transferable to the ESF-20/64 storage via BASIC programming.

The end-user, single-unit price is $199.50. Starter kits, available for an additional $25, include the user's manual and ten 20-foot wafers (tape car-



Exatron's stringy floppy mass storage peripheral for Commodore 64 and VIC.

tridges). The cost of individual wafers is about $3.

Exatron
181 Commercial Street
Sunnyvale, CA 94086
(408)737-7111
or outside CA (800)538-8559

Full-stroke Keyboard For The Atari 400

Inhome Software Incorporated has announced the B Key 400 full-stroke keyboard for the Atari 400 computer as an option to the existing membrane keyboard.

This new keyboard for the Atari 400 computer provides home computer users with all of the features of the full-stroke keyboard.

Inhome Software
2485 Dunwin Drive
Mississauga, Ontario L5L 1T1, Canada
(416) 828-0775

Inhome Software's B Key 400.

DB Master Accessory For The Apple

A statistical software package for *DB Master* users is being introduced by Stoneware, Inc. *DB Master Stat Pak* is an accessory for the *DB Master* data base management program for the Apple II or Apple II Plus.

The *Stat Pak* permits statistical analysis of data contained in *DB Master* files. *Stat Pak* is compatible with both the *DB Master* standard and Special Edition for hard disk systems.

DB Master uses values from any numeric, dollar/cents or computed fields. It also performs tests on selected records in a file, including: mean, standard deviation and standard error, co-efficient of variation, frequency of distribution, un-paired t-test, Mann Whitney U-test, Wilcoxen Paired Sample Test, linear regression, correlation and one-way analysis of variation (ANOVA) with Newman-Keuls Test and Chi Square Test (Chi Square can use alphanumeric data).

The hardware requirements are a 48K Apple or Apple II Plus with a minimum of one disk drive. The introductory price for the new *DB Master Stat Pak* is $99.

Stoneware, Inc.
50 Belvedere Street
San Rafael, CA 94901
(415)454-6500

Shapes In Color For The Apple

Shapes in Color is a BASIC precision shape-drawing program for the Apple II that can be used by