Applications And Utilities *Estimating TI Memory, Sound On The Sinclair/Timex, Applesoft Printer Control, And Many More*

COMPUTE!

$2.50
April
1983
Issue 35
Vol. 5, No. 4
63379 £1.85 in UK

The Leading Magazine Of Home, Educational, And Recreational Computing

Air Defense:
An Exciting Game Program For VIC-20, Atari, TI-99/4A, TRS-80 Color Computer, Apple, And PET/CBM

Ready To Use Word Processing Programs For VIC-20 And Atari

VIDEO-80: 80 Columns On Your Atari Via Software

Dr. Video: Enhanced Screen Utilities For VIC-20, 64, And PET/CBM

Apple Bar Charts And Many Other Programs

Buying The Right Word Processor
Plus Reviews, New Products, And More

0 74470 63379 04

April 1983 Vol. 5, No. 4

FEATURES

EDUCATION AND RECREATION

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

The Color Computer and TI-99/4A data base programs scheduled for this issue will appear next month.

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 505 Edwardia Drive, Greensboro, NC 27409. 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.

EDITOR'S NOTES

The industry price blitz continues with VIC recently advertised at K-Mart for $139, Atari 400 falling below $200, and the Commodore 64 beginning to approach the $400+ price point. Should you wait?

After all, remember when a calculator with *memory* was only $79.95 at Sears? A brokerage house acquaintance recently remarked on the falling prices (he purchased his home computer six months ago for half again as much as it now sells for) by commenting whimsically on the prices, but concluding that he was glad he'd gone ahead and started when he did. He felt as though he was those six months further along in the personal computer revolution.

It will be interesting to see the effect of Atari's recently announced computer/keyboard upgrade for the Atari 2600 game machine. While the price of the unit is expected to be around $90 (we expect the VIC-20 to be less than $100 soon), the Atari unit does, undeniably, have an installed base of millions of potential game machines.

The bottom line, of course, is that all of these new products, price cuts, expansions, and the like simply help make the consumer computer marketplace a mass market reality that much faster. A year or so ago, we did a series on the fine art of raising funds for the purchase of microcomputers for schools. We'd like to update those suggestions and helpful hints with more recent information. After all, a year or two in this industry is a long time, and fund-raising strategies for $1200 machines are vastly different from those for $200 ones. Have you found that school systems are more likely now to provide funding directly? Have you found that parent organizations are more involved? We'd like to present the collective wealth of tactics used by you readers active in educational support. So drop us a one or two page note about your successes and plans, and we'll put them together in an upcoming issue.

In deference to potential problems with confusion of names, we've retitled our newest publication *COMPUTE!'s Commodore Gazette*. This should prevent any confusion with the quarterly magazine produced by Commodore and called *Commodore*. We've also pulled our release issue date for the new monthly to June 1983. I'm pleased to announce that Tom Halfhill of our staff, who many of you have come to appreciate as Features Editor of **COMPUTE!**, will be serving as Acting Editor of *COMPUTE!'s Commodore Gazette*.

Our **COMPUTE! Books** Division is currently undergoing substantial expansion as well. If you're presently working on a title or titles in the consumer computer end of the marketplace, we'd be interested in talking with you. Please contact Scott Card, Senior Editor, Book Division at our home office. Our first titles for the Texas Instruments personal computer and the Radio Shack Color Computer will be released soon. Our thanks to you authors who have started contributing applications articles and materials to **COMPUTE!**.

We're currently investigating the possibilities for delivering portions of our printed software in machine readable form. **COMPUTE!** currently publishes more software in each issue than any magazine in the industry, and we're aware that some mechanism for electronic delivery might be helpful to our readers.

The variety of options range from direct sale of tapes and disks to resource centers such as CompuServe and The Source. We'd like your thoughts and suggestions as well. Short comments can be directed to us on the Editor's Feedback card in the back of the magazine. If you need more room, please feel free to write us a letter. As always, your thoughts and input are invaluable to us.

Robert Lock

READERS' FEEDBACK

The Editors and Readers of COMPUTE!

Make A TV Into A Monitor?

I have heard that it is possible to improve the picture quality of a computer output to a television by rewiring the TV as a monitor in some way. Would you outline the differences between a TV and a monitor for me? Is it possible and practical to convert a TV into a monitor?

Charles Coleman

It's possible, but not practical. You can bypass the receiver section of a television and route input directly to the video stages. However, this is an extremely unsafe practice. Contact with the voltages present inside a TV is likely to cause more than just an unpleasant tingle, and since many sets have a "hot" chassis it is difficult to isolate these voltages from your computer. Why risk ruining your television and computer (and possibly yourself) when a true monitor costs no more than a regular TV?

Atari Revision B

I have had an opportunity recently to upgrade my computer to a "new" version of Atari. Little did I realize at the time that a new version of the operating system was incorporated in the new Atari, rendering a large portion of my available software useless (e.g., *Ghost Hunters* by Arcade Plus).

Can other ways be devised to load the software other than via the operating system? Help?

G. Smyczynski

A few pieces of commercial software will not run on the Revision B Operating System (OS) due to illegal OS calls. Contact the software companies with regard to any updates. If you can acquire an extra 10K ROM board, you can choose either operating system (on the Atari 800 only) merely by changing boards.

Translating Programs For The TI

I own a TI-99/4A home computer. I like your magazine, but I have tried and cannot convert the programs in **COMPUTE!** to run on the TI. Especially hard to figure out are the PEEK and POKE statements.

Could you please explain how to convert the programs to TI? I do appreciate your new TI. column.

John Dobrinski

Texas Instruments appears to have developed their BASIC from a slightly different perspective than many other microcomputer BASICs. The PEEK and POKE commands allow programmers to examine and modify individual memory locations. While this may be a desirable feature on a personal computer, it could be undesirable on a large, multi-user system so no "mainframe" BASICs support these operations. TI BASIC shares this feature of minicomputer BASICs.

Fortunately, TI substitutes an impressive "library" of built-in ROM subroutines which accomplish most of the same things that PEEK and POKE are used for on other computers. For beginners this may even be an advantage, since the subroutine CALLs are usually more easily understood than the equivalent PEEKs and POKEs. For example, to read the TI joysticks you can type:

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

Contrast this with the equivalent for the VIC-20:

```
100 POKE 37154,127: X=(NOTPEEK(37151))AND 60 -
    ((PEEK(37152) AND 128)=0) : POKE 37154,255
```

Other impressive features are CALL CHAR, RESEQUENCE, and NUMBER. These provide built-in character definition, renumber, and automatic line numbering utilities.

Should I Buy A Computer?

Recently, my interest has turned toward finding out about home computers. As a start, I purchased **COMPUTE!**, and visited a few stores with home computers and software.

So far, my observations show that the home computer market is directed mostly toward games, especially space and war games.

Since there is an eleven year old in the family, my interest is also in the educational aspect of home computers.

I have two basic concerns before I spend hundreds of dollars and find that a home computer is not a waste of money, a flash in the pan, or a pie-in-the-sky promise. Please advise me where local sales personnel and even some teachers are not able to answer me:

Air Defense

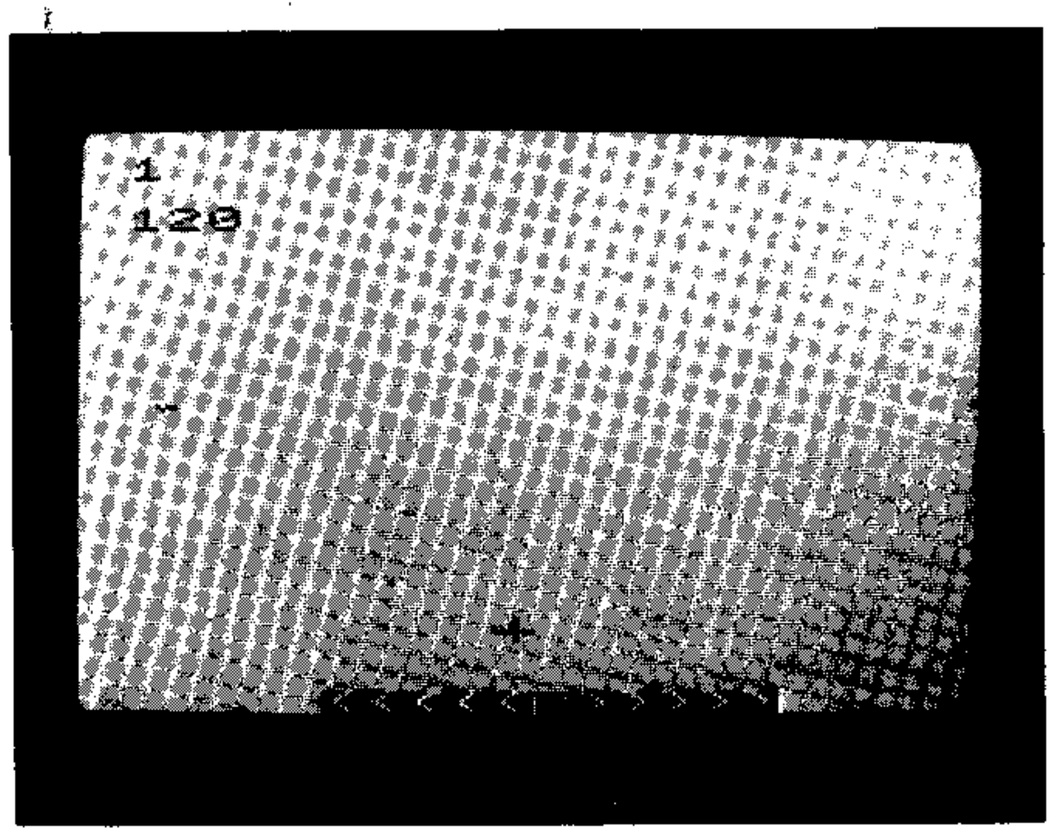
T. L. Wahl

"Air Defense" is a challenging game for the 5K VIC-20, 16K Atari 400/800, unexpanded TRS-80 Color Computer, Apple II, TI-99/4A, and PET/CBM. Look in the article for special notes on your particular machine.

The object of the game of "Air Defense" is to defend your land (at the bottom of the screen) from falling bombs. The bombs appear at various places at the top of the screen. As they fall, the player must line up the crosshair of his gunsight and fire when the bomb and crosshair are aligned. On the VIC version press S to move up, X to move down, ‹cursor down› to move left, and ‹cursor up› to move right. Press SPACE to fire.

The player gets only one shot, and timing is critical. After 20 bombs have appeared, the game ends, and the player is given a score showing hits and misses and a point score.

One of the unique features of the game is the increasing difficulty factor: as the player improves his skill, the crosshair is gradually moved toward the top of the screen, and quicker reflexes and improved technique are required to destroy the falling bombs. As a reward for increasing skill, the player earns higher point values for successive hits. In addition, the player receives a higher score the sooner the falling bomb is destroyed.



A bomb explodes in the VIC-20 version of "Air Defense" (PET/CBM and Apple versions are similar).

Program 1: VIC Version

```
100 X=RND(0)
110 A=8152:B=38872:P=0:M=0:T=0:Q=0
120 PRINT"{CLEAR}{07 DOWN}     AIR DEFENSE
    "
130 PRINT"{02 DOWN}     DO YOU NEED"
140 PRINT"{DOWN}    INSTRUCTIONS?"
150 PRINT"{DOWN}   TYPE 'Y' OR 'N'"
160 FOR H=1TO1000:GETD$
170 IF D$="N" THEN 380
180 IF D$="Y" THEN 220
190 NEXT
200 PRINT"{CLEAR}{DOWN}YOU DID NOT PRESS '
    Y' OR 'N'."
210 FOR K=1TO5000:NEXT:GOTO120
220 PRINT"{CLEAR}  YOU MUST STOP THE"
230 PRINT"  FALLING BOMB BY"
240 PRINT"   EXPLODING IT IN"
250 PRINT"     MID-AIR."
260 PRINT"{DOWN} MOVE THE CROSSHAIR"
270 PRINT"{DOWN}*{REV}LEFT{OFF}:CURSOR U/D
     KEY"
280 PRINT"{DOWN}*{REV}RIGHT{OFF}:CURSOR L/
    R KEY"
290 PRINT"{DOWN}*{REV}UP{OFF}:WITH THE 'S'
     KEY"
300 PRINT"{DOWN}*{REV}DOWN{OFF}:WITH THE '
    X' KEY"
310 PRINT"WHEN THE BOMB AND THE"
320 PRINT"CROSSHAIR ARE LINED UP, FIRE BY ~
    PRESSING THESPACE";
330 PRINT" BAR."
340 PRINT"{DOWN}PRESS ANY KEY TO START"
350 GET D$:IF D$="" THEN 350
360 PRINT"{CLEAR}{10 DOWN}      GOOD LUCK!
    "
370 FOR I=1TO2500 :NEXT
380 IFT=20 THEN 860
390 PRINT"{CLEAR}":D=INT(RND(1)*10)
400 T=T+1
410 E=D+7685
420 F=D+38405
430 PRINTP*Q*10
440 FOR I=1 TO 200:NEXTI
450 POKE A,91:POKE B,0
460 GET A$
470 IFA$="S"THENA=A-22:B=B-22
480 IF A$="X"THEN B=B+22:A=A+22
```

```
290  PRINT : PRINT : PRINT " WHEN THE B
     OMB AND THE CROSSHAIR ARE IN": PRINT
     " LINE, FIRE BY PRESSING THE SPACE
      BAR."
300  FOR I = 1 TO 10000: NEXT : HOME : VTAB
     10: HTAB 15: FLASH : PRINT "GOOD L
     UCK!": FOR I = 1 TO 5000: NEXT : NORMAL

310 VC = 22
320  IF T = 20 THEN 780
330  HOME : VTAB 24: INVERSE : FOR I =
     2 TO 39: HTAB I: PRINT " ";: NEXT
     I: NORMAL
340 HC = 21:T = T + 1:VB = 0: VTAB 2: HTAB
     3: PRINT T
350  VTAB 4: INVERSE : PRINT P * Q * 10
     : NORMAL
360 HB =  INT ( RND (1) * 29) + 6
370 VB = VB + 1
380  IF VB = 1 THEN 400
390  HTAB HB: VTAB VB - 1: PRINT " "
400 OVCROSS = VC:OHCROSS = HC
410 A =  PEEK ( - 16384): POKE  - 16368
     ,0
420  IF A - 128 =  ASC ("S") THEN VC =
     VC -  SGN (VC - 1)
430  IF A - 128 =  ASC ("X") THEN VC =
     VC +  SGN (22 - VC)
440  IF A = 136 THEN HC = HC -  SGN (HC
      - 2)
450  IF A = 149 THEN HC = HC +  SGN (39
      - HC)
460  IF VC = OVCROSS AND HC = OHCROSS THEN
     480
470  HTAB OHCROSS: VTAB OVCROSS: PRINT
     " "
480  HTAB HC: VTAB VC: PRINT "+"
490  HTAB HB: VTAB VB: PRINT "*"
500  FOR I = 1 TO 50: NEXT I
510  IF VB = 23 THEN 560
520  IF VB = VC AND HB = HC THEN 540
530  GOTO 370
540  IF A - 128 =  ASC (" ") THEN 650
550  GOTO 370
560  REM  MISS
570  VTAB VB: HTAB HB: PRINT " "
580  VTAB 24: INVERSE : FOR I = 1 TO 5:
      HTAB HB - I: PRINT "<";: HTAB HB +
     I: PRINT ">";: NEXT I
590  FOR K = 1 TO 100
600  CALL 768
610  NEXT K
620  FOR I = 1 TO 100: NEXT I
630 M = M + 1: GOTO 320
640  REM  HIT
650  HTAB HC - 1: VTAB VC - 1: PRINT  CHR$
     (220);" /"
660  HTAB HC - 1: VTAB VC: PRINT "- -"
670  HTAB HC - 1: VTAB VC + 1: PRINT "/
      "; CHR$ (220)
680  REM  SOUND ROUTINE
690  FOR K = 1 TO 20
700  FOR I = 1 TO K
710  CALL 768
720  NEXT I
730  NEXT K
740 Q = Q + 23 - VC
750 P = P + 1
760 VC = VC - 1
770  GOTO 320
780  REM  GAME OVER
790  HOME
800  VTAB 8: HTAB 15: FLASH : PRINT "GA
     ME OVER": NORMAL
810  VTAB 12: PRINT  TAB( 12)"DESTROYED
      "P
820  VTAB 14: PRINT  TAB( 15)"MISSED "M

830  VTAB 17: PRINT  TAB( 11)"YOUR SCOR
     E "P * Q * 10
840  VTAB 20: HTAB 10: INPUT "ANOTHER G
     AME (Y/N) ";AN$
850  IF AN$ = "N" THEN 880
860  IF AN$ <  > "Y" THEN  VTAB 20: HTAB
     29: PRINT "      ": GOTO 840
870  RUN
880  END
```

TI-99/4A Notes

The TI-99/4A version of Air Defense is similar to the VIC-20 version. In fact, scoring is calculated in the same manner: the sooner the bombs are destroyed, the higher the score. However, the TI-99/4A version's graphics are drawn with custom characters.

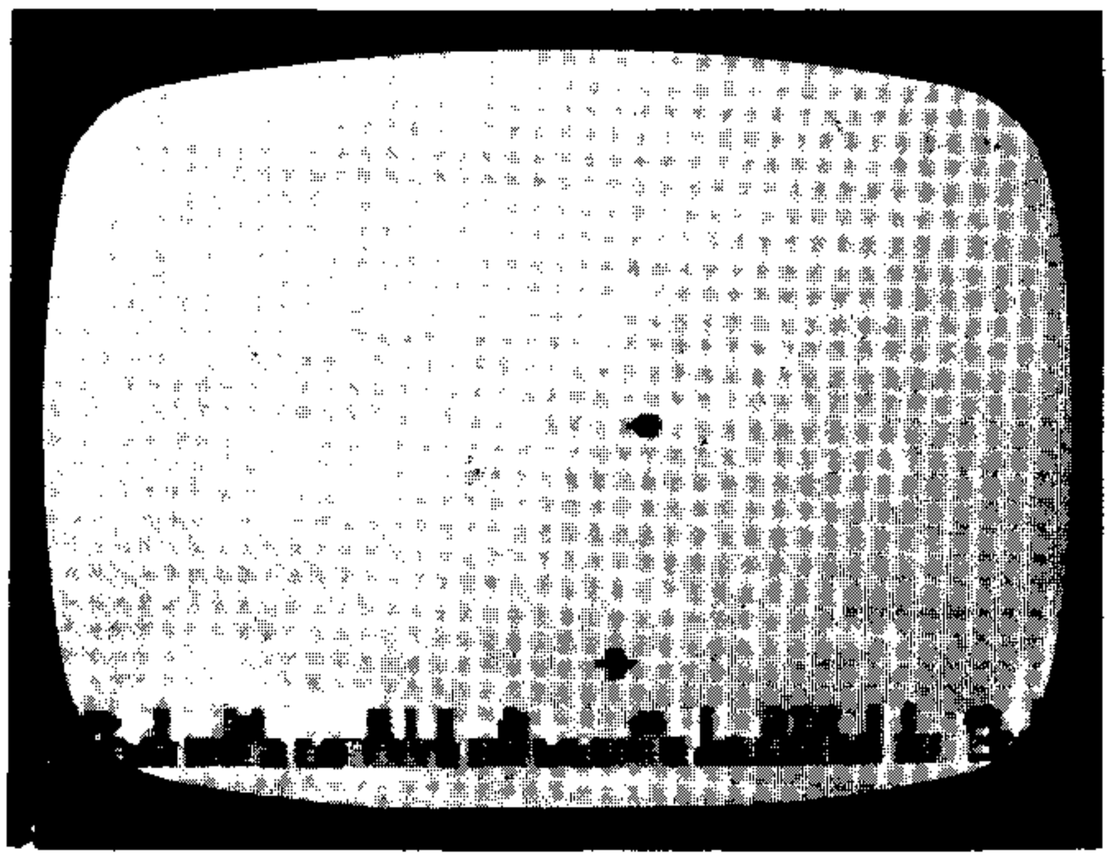
Most of the shapes in the game are custom characters that were designed with the aid of the character definition program in the TI-99/4A *User's Reference Guide* (pages III-26 and III-27). Custom characters created in this manner were then assigned ASCII code numbers in the range 122-136, which correspond to character sets 12, 13, and 14. Since no character set higher than 14 is referenced in the program, the Extended BASIC mode can be used for a faster, more challenging game.

Program 5: TI Version

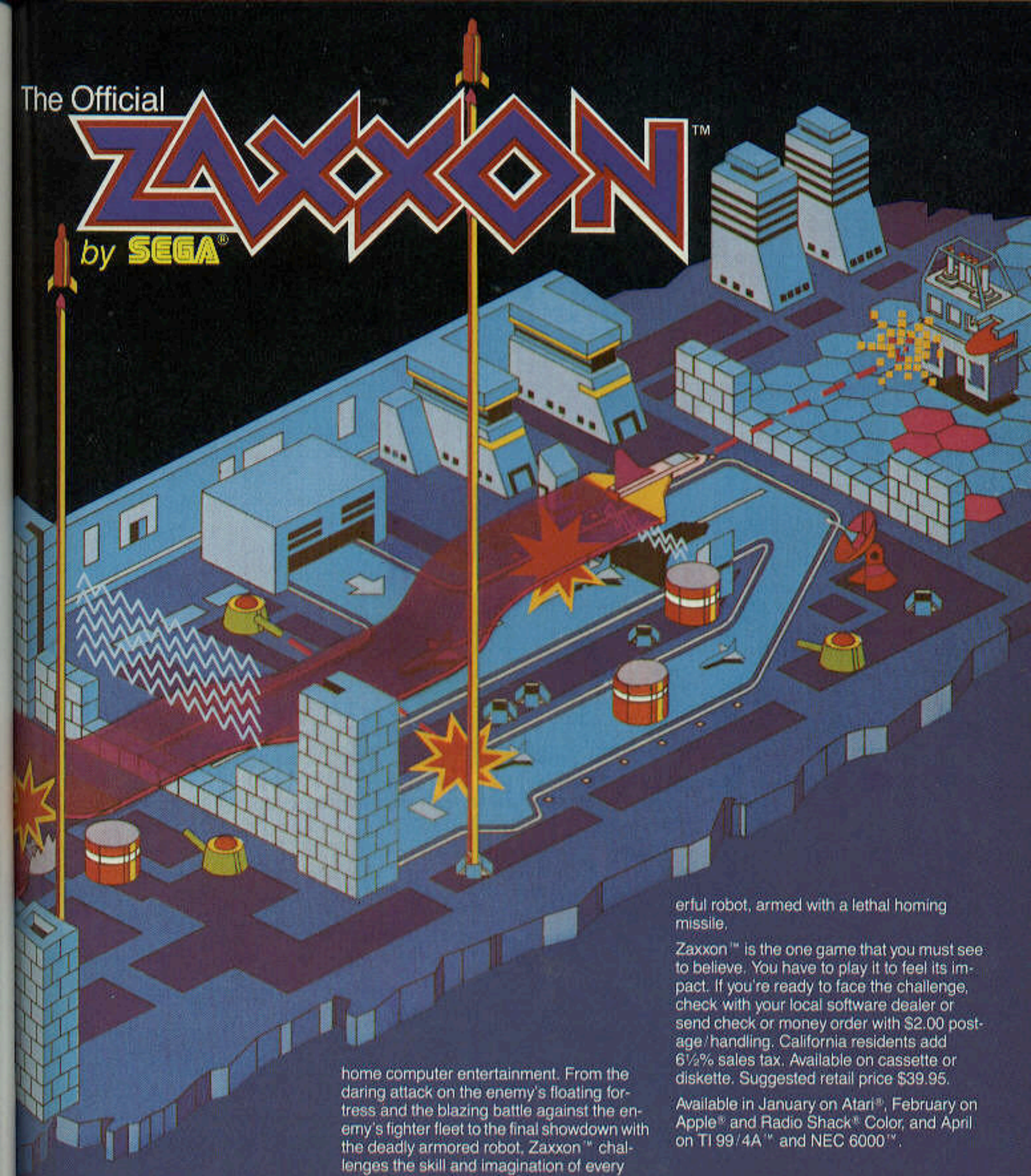
```
100 DIM BLOCK$(2),PLACE(2),BUILDING(
    32,2)
110 RANDOMIZE
120 REM BOMB CHARACTER
130 CALL CHAR(129,"001CBEFFFFBE1C00"
    )
140 REM CROSSHAIR CHARACTER
150 CALL CHAR(130,"181818FFFF181818"
    )
160 CALL CLEAR
170 CALL SCREEN(12)
180 FOR J=5 TO 8
190 CALL COLOR(J,5,16)
200 NEXT J
210 FOR J=9 TO 12
220 CALL COLOR(J,2,14)
230 NEXT J
240 T=0
250 P=0
260 Q=0
270 M=0
280 CALL CLEAR
290 PRINT "{8 SPACES}AIR DEFENSE"
300 PRINT
310 PRINT
320 PRINT
330 PRINT " do you need instructions
    ?"
340 PRINT
```

```
350 PRINT "{8 SPACES}type Y or N"
360 FOR I=1 TO 7
370 PRINT
380 NEXT I
390 CALL KEY(3,Y,STATUS)
400 IF STATUS=0 THEN 390
410 IF Y=ASC("N")THEN 750
420 IF Y=ASC("Y")THEN 520
430 CALL CLEAR
440 PRINT
450 PRINT "  you did not press Y or
    N."
460 FOR I=1 TO 13
470 PRINT
480 NEXT I
490 FOR DELAY=1 TO 500
500 NEXT DELAY
510 GOTO 280
520 CALL CLEAR
530 PRINT "{3 SPACES}YOU MUST STOP T
    HE FALLING"
540 PRINT "BOMB BY EXPLODING IT IN M
    ID-AIR."
550 PRINT
560 PRINT
570 PRINT "{3 SPACES}-MOVE THE CROSS
    HAIR-"
580 PRINT
590 PRINT "  left :HOLD THE s KEY"
600 PRINT "  right:HOLD THE d KEY"
610 PRINT "  up{3 SPACES}:HOLD THE e
     KEY"
620 PRINT "  down :HOLD THE x KEY"
630 PRINT
640 PRINT "{3 SPACES}WHEN THE BOMB A
    ND THE"
650 PRINT "CROSSHAIR ARE LINED UP,"
660 PRINT "FIRE BY PRESSING THE SPAC
    E"
670 PRINT "BAR. THE SOONER YOU GET T
    HE"
680 PRINT "BOMB, THE HIGHER YOUR SCO
    RE."
690 PRINT
700 PRINT
710 PRINT
720 PRINT "{3 SPACES}PRESS any key T
    O START"
730 CALL KEY(0,S,STATUS)
740 IF STATUS=0 THEN 730
750 CALL CLEAR
760 CALL COLOR(8,2,1)
770 PRINT "{7 SPACES}GOOD LUCK!!!"
780 FOR I=1 TO 10
790 PRINT
800 NEXT I
810 IF R=ASC("R")THEN 840
820 GOSUB 2090
830 GOTO 860
840 FOR I=1 TO 250
850 NEXT I
860 CALL CLEAR
870 GOSUB 2300
880 IF T=20 THEN 1860
890 T=T+1
900 CCROSS=16
910 RCROSS=21
920 RBOMB=1
930 CALL SCREEN(6)
940 CBOMB=INT(RND*29)+2
950 H$=STR$(T)
960 ROW=2
970 COL=3
980 GOSUB 2520
990 SCORE=P*Q*10
1000 H$=STR$(SCORE)
1010 ROW=5
1020 GOSUB 2520
1030 FOR I=1 TO 70
1040 NEXT I
1050 FOR I=2 TO 5 STEP 3
1060 CALL HCHAR(I,3,32,6)
1070 NEXT I
1080 OLDRCROSS=RCROSS
1090 OLDCCROSS=CCROSS
1100 CALL KEY(0,A,STATUS)
1110 IF A<>ASC("E")THEN 1130
1120 RCROSS=RCROSS-SGN(RCROSS-1)
1130 IF A<>ASC("X")THEN 1150
1140 RCROSS=RCROSS+SGN(22-RCROSS)
1150 IF A<>ASC("D")THEN 1170
1160 CCROSS=CCROSS+SGN(31-CCROSS)
1170 IF A<>ASC("S")THEN 1190
1180 CCROSS=CCROSS-SGN(CCROSS-2)
1190 IF RBOMB=1 THEN 1210
1200 CALL VCHAR(RBOMB-1,CBOMB,32)
1210 IF (RCROSS=OLDRCROSS)*(CCROSS=O
     LDCCROSS)THEN 1230
1220 CALL VCHAR(OLDRCROSS,OLDCCROSS,
     32)
1230 CALL VCHAR(RCROSS,CCROSS,130)
1240 CALL VCHAR(RBOMB,CBOMB,129)
1250 RBOMB=RBOMB+1
1260 IF RBOMB=23 THEN 1540
1270 IF (RCROSS=RBOMB-1)*(CCROSS=CBO
     MB)THEN 1290
1280 GOTO 1080
1290 CALL KEY(0,B,STATUS)
1300 IF B=32 THEN 1330
1310 GOTO 1080
1320 REM BOMB DESTROYED
1330 RBOMB=RBOMB-1
1340 CALL SCREEN(10)
1350 CALL VCHAR(RBOMB,CBOMB,32)
1360 CNT=0
1370 C1=92
1380 C2=47
1390 FOR I=-1 TO 1 STEP 2
1400 CALL VCHAR(RBOMB+I,CBOMB+I,C1)
1410 CALL VCHAR(RBOMB+I,CBOMB-I,C2)
1420 NEXT I
1430 C1=32
1440 C2=32
1450 IF CNT=1 THEN 1510
1460 CNT=1
1470 FOR VOL=10 TO 30 STEP 5
1480 CALL SOUND(100,-6,VOL)
1490 NEXT VOL
1500 GOTO 1390
1510 P=P+1
1520 Q=Q+(23-RBOMB)
1530 GOTO 880
1540 REM BOMB HITS THE CITY
1550 CALL VCHAR(22,CBOMB,32)
1560 CALL SCREEN(9)
1570 CALL COLOR(12,11,1)
1580 CALL VCHAR(23,CBOMB-1,122)
1590 CALL VCHAR(23,CBOMB,32)
1600 CALL VCHAR(23,CBOMB+1,123)
1610 CALL VCHAR(24,CBOMB-1,124)
1620 CALL VCHAR(24,CBOMB,125)
1630 CALL VCHAR(24,CBOMB+1,126)
```

```
1640 FOR I=1 TO 20
1650 NEXT I
1660 CALL COLOR(12,7,1)
1670 CALL SCREEN(12)
1680 FOR I=1 TO 20
1690 NEXT I
1700 CALL SCREEN(7)
1710 FOR VOL=24 TO 1 STEP 4
1720 CALL SOUND(200,-7,VOL)
1730 NEXT VOL
1740 FOR DVOL=1 TO 24 STEP 4
1750 CALL SOUND(200,-7,DVOL)
1760 NEXT DVOL
1770 FOR J=23 TO 24
1780 FOR I=CBOMB-1 TO CBOMB+1
1790 CALL VCHAR(J,I,32)
1800 NEXT I
1810 NEXT J
1820 CALL VCHAR(RCROSS,CCROSS,32)
1830 CALL COLOR(12,2,14)
1840 M=M+1
1850 GOTO 880
1860 CALL CLEAR
1870 CALL SCREEN(4)
1880 CALL COLOR(8,5,16)
1890 PRINT "{9 SPACES}GAME OVER"
1900 FOR I=1 TO 4
1910 PRINT
1920 NEXT I
1930 PRINT "{3 SPACES}DESTROYED
     {3 SPACES}";P
1940 PRINT
1950 F[0{,}!1{5 ,}"{3 SPACES}MISSED
     {6 SPACES}";M
1960 PRINT
1970 PRINT "{3 SPACES}TOTAL POINTS";
     P*Q*10
1980 FOR I=1 TO 4
1990 PRINT
2000 NEXT I
2010 PRINT "{3 SPACES}PRESS r TO PLA
     Y AGAIN"
2020 PRINT
2030 PRINT
2040 CALL KEY(0,R,STATUS)
2050 IF STATUS=0 THEN 2040
2060 IF R=ASC("R")THEN 160
2070 END
2080 REM READ CITY DATA
2090 FOR ROW=2 TO 1 STEP -1
2100 FOR COL=1 TO 32
2110 READ BUILDING(COL,ROW)
2120 NEXT COL
2130 NEXT ROW
2140 REM CUSTOM CHAR & COLORS
2150 CALL CHAR(136,"FFABFFABFFABFFFF
     ")
2160 CALL CHAR(128,"003C7EFFFFFF7E42
     ")
2170 CALL CHAR(131,"42665A6642427E66
     ")
2180 CALL CHAR(132,"606060606060606 0
     ")
2190 CALL CHAR(133,"607858F8D8F8D8F8
     ")
2200 CALL CHAR(134,"F8A8F8A8F8A8F8F8
     ")
2210 CALL CHAR(135,"C3C3FFABFFABFFF
     ")
2220 CALL COLOR(14,7,12)
2230 CALL CHAR(122,"8040201008040201
     ")
2240 CALL CHAR(123,"0102040810204080
     ")
2250 CALL CHAR(124,"80E0F8FEFFFFFFFF
     ")
2260 CALL CHAR(125,"8142241800 81C3E7
     ")
2270 CALL CHAR(126,"01071F7FFFFFFFFF
     ")
2280 RETURN
2290 REM SET UP CITY
2300 FOR ROW=2 TO 1 STEP -1
2310 FOR COL=1 TO 32
2320 BLOCK$(ROW)=BLOCK$(ROW)&CHR$(BU
     ILDING(COL,ROW))
2330 NEXT COL
2340 NEXT ROW
2350 FOR ROW=2 TO 1 STEP -1
2360 FOR COL=1 TO 32
2370 PLACE(ROW)=ASC(SEG$(BLOCK$(ROW)
     ,COL,1))
2380 CALL HCHAR(ROW+22,COL,PLACE(ROW
     ))
2390 NEXT COL
2400 NEXT ROW
2410 RETURN
2420 REM CITY DATA
2430 DATA 136,134,131,135,133,136,13
     6,133
2440 DATA 135,136,136,136,133,136,13
     6,135
2450 DATA 135,136,136,134,133,136,13
     6,136
2460 DATA 135,132,136,32,131,135,132
     ,135
2470 DATA 134,133,128,32,132,32,135,
     32
2480 DATA 32,32,134,132,132,32,133,3
     2
2490 DATA 32,32,128,32,132,32,133,13
     5
2500 DATA 32,132,132,32,128,32,132,3
     2
2510 REM HORIZONTAL # PRINTER
2520 FOR I=1 TO LEN(H$)
2530 DIGIT=ASC(SEG$(H$,I,1))
2540 CALL HCHAR(ROW,COL+I,DIGIT)
2550 NEXT I
2560 RETURN
```



he crosshair stands ready to intercept a bomb descending ward multicolored buildings in the TI version of "Air efense."



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

Typing Teacher

Alan McCright

For Atari, VIC, TI-99/4A, and Apple. Typing in program listings is far easier if you really know the keyboard and don't need to watch your fingers. The typing program given here helps you learn the keyboard, and will give you a score based either on characters per minute or in words per minute.

Those who must rely on hunt-and-peck typing have likely discovered just how tedious it can be, especially when you are typing in programs. This program is a self-teacher that will help familiarize you with keyboard layout and help you learn to touch type.

The idea is to let your fingers find the correct key, and not to look at the keyboard. When this program is RUN, a representation of the keyboard layout appears on the screen. The characters are printed in an approximation of their keyboard positions. Check the key's location on the display, and try to get your finger to move there without looking down at the keyboard.

The Atari version starts the clock at memory location 19 and will print a non-SHIFTed character on the upper center of the screen. It will then wait for your response, flash the screen character that corresponds to your typed key, and check to see if it matches the test character. If so, your score will be incremented by one. After one minute, the test will end, your score in characters per minute will be printed, and you will be asked to try again. If you are not using the Atari version of this program, see the notes specific to your computer.

Getting the proper screen character to flash was a problem. I know of no way to read the keyboard in x,y, and a data READ after each GET was much too slow. Finally, I hit upon the idea of POKEing the screen characters' x,y positions into page six at the locations corresponding to the characters' ATASCII values times two and their values times two plus one (lines 160 and 400). Since the GET function returns the ATASCII value, a simple algorithm and a couple of PEEKs will fetch the proper screen coordinates.

The four DATA statements contain the ATASCII values of the characters in their relative keyboard positions (line 1000 = keyboard row 1).

At line 100, the screen y coordinate starts at row 5. This is incremented by one at the end of each DATA statement.

Line 120 is set to 3 at the beginning of each row, incremented by 1 after each READ, and is POKEd into memory as the screen x position.

Line 160 POKEs this data into the appropriat memory location.

Line 170 then uses these values to print the character in its proper screen position before goin to the next READ.

The rows are put on the screen beginning at column 3. The last two DATA lines are padded with spaces (32) at the beginning, to position thos rows one column over.

If you prefer to see your score in words per minute, make these changes:

```
500 POSITION 9,2: ?#6;CHARCNT/5
510 POSITION 3,3: ?#6;"WORDS PER MINUTE"
```

This assumes that the average English word is five letters long. However, since the characters are chosen at random (which I found ideal for learning to type in programs), each individual character has to be recognized rather than recalle as part of a word. Thus, scoring in words per minute will lead to some appallingly low, though accurate, scores, even for good typists.

How fast can the program run? In the word-per-minute mode, by deleting line 360 and all of the REMs, and holding down any key after RUNning, a score of 60-70 words per minute is typical However, when you are actually testing, your own reaction time will keep you from reaching that level. You might want to modify the routine using word lists instead of random characters to get an idea of your true secretarial speed.

Program 1: Atari Version

```
10 GRAPHICS 2+16
20 POSITION 2,0:? #6;"TYPING  TEACH
   R":REM INVERSE VIDEO
30 OPEN #1,4,0,"K:"
40 CHARCNT=0:REM ZERO CHARACTER COU
   TER
99 REM ** ROUTINE TO ENTER CHARACTE
    POSITION DATA **
100 FOR ROW=5 TO 8:REM ROW DATA TO
    OKE
120 FOR COL=3 TO 15:REM COLUMN DATA
    TO POKE
130 READ CHAR
```

Apple, VIC, And TI-99/4A Notes

Apple

Because the Apple lacks a realtime clock, a special counter routine must be employed in this version of the program. Incrementing occurs in line 320 while waiting for a keyboard response, and again in line 350 to account for the time required to process each response. After approximately a minute, a certain counter value will be reached (in line 330) and the testing routine will be halted and a score displayed. As in the Atari version, your score can be given in words per minute by making line 440 read:

```
440 HTAB12:VTAB7:INVERSE:PRINT"WORDS/
    MINUTE=";" ";CCNT/5:NORMAL
```

If you modify this program, be sure to check the timing for you may have affected it. If so, adjust line 350.

VIC

The VIC version of "Typing Teacher" POKEs the X,Y coordinates for each character used on the screen keyboard in an area of memory normally used as a "cassette buffer" (to hold items coming into or going out from the cassette during SAVEs or LOADs). Typing skill can be evaluated on a words per minute basis by changing line 580 to read:

```
580 PRINT"[HOME][06 DOWN][04 RIGHT]
    [REV]WORDS/MINUTE[OFF]";"=";CCNT/5
```

TI-99/4A

The TI-99/4 version, much like the Apple version, uses an incrementing counter to time the speed of keyboard response. This process occurs in lines 570 and 640. Since POKEs aren't allowed in TI BASIC, the X,Y coordinates for the characters in the keyboard displayed on the screen must be stored in an array. The TI-99/4 is somewhat slower in processing, and the sorting that is required to flash the correct keyboard response in lines 770 to 850 causes further delay. Processing speed for each keyboard response can be increased somewhat by changing line 650 to read:

```
650 IF CR<>N THEN 860
```

so that the character flashing routine on the screen-formatted keyboard is not executed. If this change is made in the program, line 640 should be changed to:

```
640 TIME=TIME+4
```

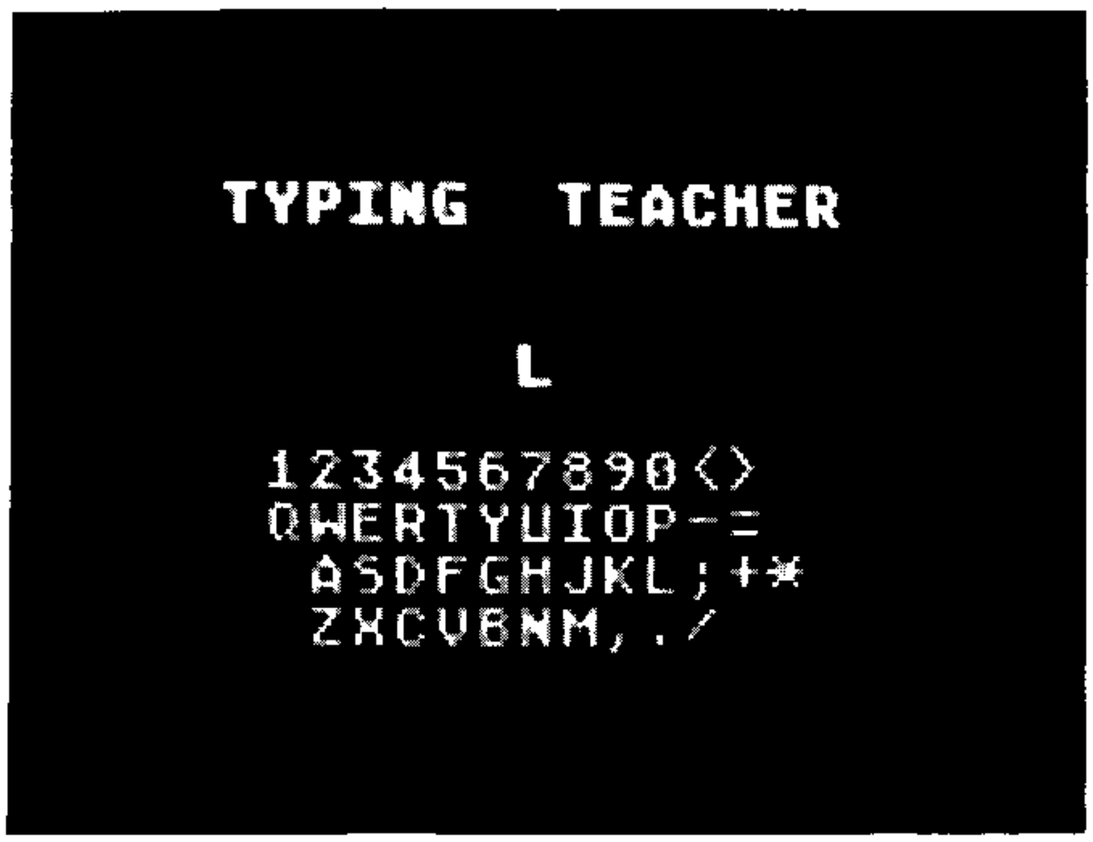
since processing time has been reduced. Unfortunately, however, you may still find yourself pushing the speed limits of the TI-99/4. Line 680 can be changed to:

```
680 PRINT" words/minute= ";CHARCNT/5
```

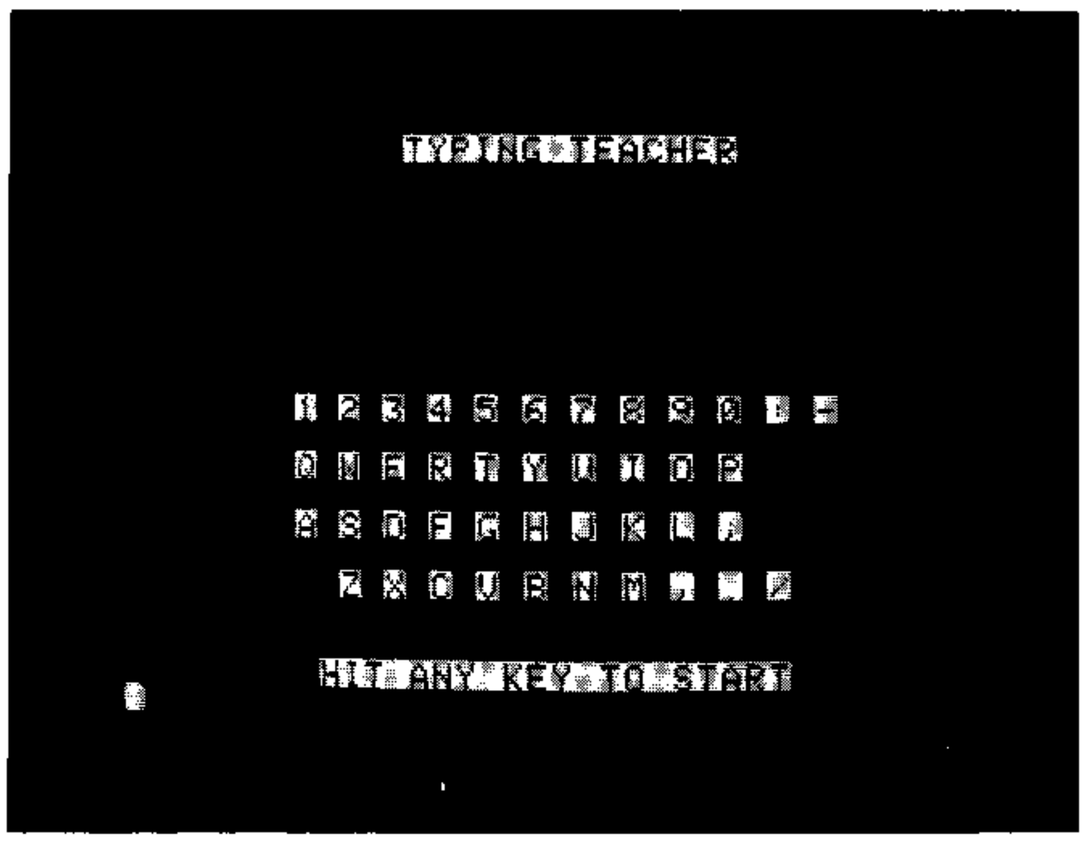
if a words per minute score is desired.

If you modify the program, see if the timing went off and make any necessary adjustments to line 640.

```
140 IF CHAR=0 THEN NEXT ROW:GOTO 120
150 IF CHAR=-1 THEN 200
160 POKE 1536+(CHAR*2),COL:POKE 1536
    +(CHAR*2)+1,ROW:REM POKE POSITIO
    N DATA
170 POSITION COL,ROW:? #6;CHR$(CHAR+
    128):REM PRINT CHAR TO SCREEN
180 NEXT COL
190 GOTO 120
199 REM **START CLOCK AND SELECT RAN
    DOM CHARACTER **
200 POSITION 2,10:? #6;"ANY KEY TO S
    TART":GET #1,CHAR:POSITION 2,10:
    ? #6;"(16 SPACES)":REM 16 SPACES
210 POKE 19,0:REM ZERO & START TIME
    COUNTER
220 N=INT((RND(0)*49)+42):REM CHOOSE
     A RANDOM CHARACTER
230 IF N=63 OR N=64 OR N=58 OR N=OLD
    CHAR THEN 220:REM IGNORE CERTAIN
     CHARACTERS
240 OLDCHAR=N
250 POSITION 9,3:? #6;CHR$(N):REM PR
    INT RANDOM NUMBER CHARACTER
260 IF PEEK(19)>=14 THEN 500:REM TIM
    E UP?
295 IF PEEK(764)=255 THEN 260
299 REM **PROCESS YOUR RESPONSE**
300 GET #1,CHAR
305 SOUND 0,10,10,8:CHARCNT=CHARCNT+
    1:REM ADD ONE TO TOTAL
310 SOUND 0,0,0,0:GOSUB 400
320 ? #6;CHR$(CHAR):REM FLASH CHARAC
    TER...
330 FOR X=1 TO 10:NEXT X
340 GOSUB 400
350 ? #6;CHR$(CHAR+128):REM ...AND R
    ETURN TO NORMAL
360 IF CHAR<>N THEN SOUND 0,150,12,8
    :FOR X=1 TO 10:NEXT X:SOUND 0,0,
    0,0:CHARCNT=CHARCNT-1:REM YOU ER
    RED
370 GOTO 220
399 REM ** POSITION CURSOR OVER TYPE
    D CHARACTER **
400 TRAP 360:POSITION PEEK(1536+(CHA
    R*2)),PEEK(1536+(CHAR*2)+1)
410 RETURN
499 REM ** CALCULATE AND PRINT SCORE
     **
500 POSITION 9,2:? #6;CHARCNT
510 POSITION 1,3:? #6;"CHARACTERS/MI
```



The Atari version of "Typing Teacher" uses large-size GRAPHICS 2 characters.



"Typing Teacher," Apple version. (TI-99/4A version similar.)

```
    NUTE"
520 POSITION 1,10:? #6;"HIT 'R' TO R
    ESTART"
530 FOR SND=1 TO 5
540 SOUND 0,30,10,8
550 FOR DELAY=1 TO 50:NEXT DELAY
560 SOUND 0,0,0,0:NEXT SND
570 IF SND<5 THEN 540
580 GET #1,RESTART:IF RESTART=ASC("R
    ") THEN RUN
700 END
999 REM ** ATASCII DATA IN INDIVIDUA
    L KEYBOARD ROWS AND COLUMNS **
1000 DATA 49,50,51,52,53,54,55,56,57
     ,48,60,62,0
1010 DATA 81,87,69,82,84,89,85,73,79
     ,80,45,61,0
1020 DATA 32,65,83,68,70,71,72,74,75
     ,76,59,43,42,0
1030 DATA 32,90,88,67,86,66,78,77,44
     ,46,47,-1
```

Program 2: Apple Version

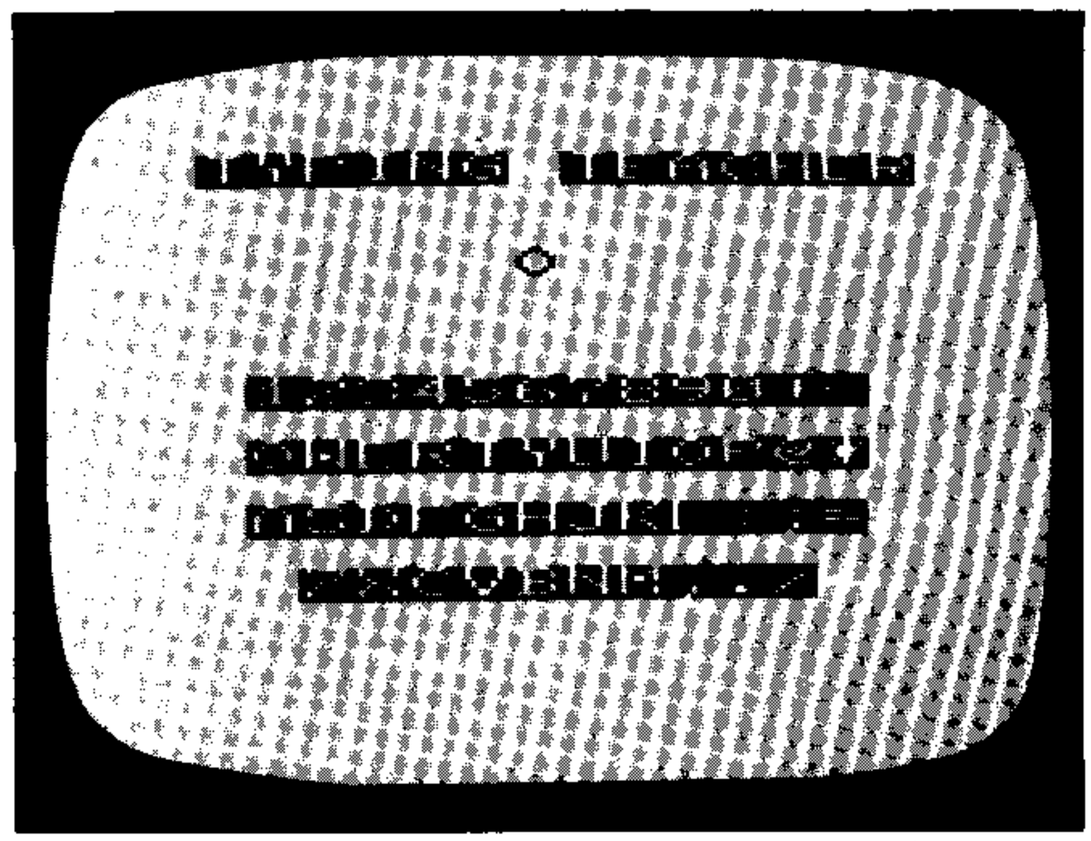
```
100  FOR I = 770 TO 795: READ M: POKE I
     ,M: NEXT
110  HOME : PRINT : HTAB 14: INVERSE :
     PRINT "TYPING TEACHER": NORMAL
120 CCNT = 0: REM  ZERO CHARACTER COUNT
     ER
130  REM  ** ROUTINE TO ENTER CHARACTER
      POSITION DATA **
140  FOR ROW = 11 TO 17 STEP 2: REM  RO
     W DATA TO POKE
150  FOR COL = 9 TO 33 STEP 2: REM   CO
     LUMN DATA TO POKE
160  READ CHAR
170  IF CHAR = 0 THEN  NEXT ROW: GOTO 1
     50
180  IF CHAR =  - 1 THEN 250
190  POKE 796 + (CHAR * 2),COL: POKE 79
     6 + (CHAR * 2) + 1,ROW
200  IF CHAR = 32 THEN 220
210  HTAB COL: VTAB ROW: INVERSE : PRINT
      CHR$ (CHAR): NORMAL
220  NEXT COL
230  GOTO 150
240  REM  ** TIMER AND SELECT RANDOM CH
     ARACTER **
250  HTAB 10: VTAB 20: INVERSE : PRINT
     "HIT ANY KEY TO START": NORMAL : GET
     A$
260  HTAB 10: VTAB 20: FOR I = 1 TO 20:
      PRINT " ";: NEXT I
270 N =  INT (( RND (1) * 47) + 44): REM
      CHOOSE A RANDOM CHARACTER
280  IF N >  = 60 AND N <  = 64 OR N =
     OLDCHAR THEN 270
290 OLDCHAR = N
300  HTAB 20: VTAB 7: PRINT  CHR$ (N): REM
      PRINT RANDOM NUMBER CHARACTER
310  REM  **PROCESS YOUR RESPONSE**
320  IF  PEEK ( - 16384) < 128 AND TIME
      < 2710 THEN TIME = TIME + 1: GOTO
     320
330  IF TIME >  = 2710 THEN 440
340  GET A$:CHAR =  ASC (A$): POKE 768,
     30: POKE 769,1: CALL 770:CCNT = CC
     NT + 1: REM  *ADD ONE TO TOTAL*
350 TIME = TIME + 10
360  GOSUB 420
370  PRINT  CHR$ (CHAR)
380  FOR I = 1 TO 10: NEXT I
390  GOSUB 420: INVERSE : PRINT  CHR$ (
     CHAR): NORMAL
400  IF CHAR <  > N THEN CCNT = CCNT -
     1: POKE 768,1: POKE 769,175: CALL
     770
410  GOTO 270
420  IF CHAR <  > N THEN  POP : GOTO 40
     0
430  HTAB ( PEEK (796 + 2 * CHAR)): VTAB
     ( PEEK (797 + 2 * CHAR)): RETURN
440  HTAB 9: VTAB 7: INVERSE : PRINT "C
     HARACTERS/MINUTE =";" ";CCNT: NORMAL
450  HTAB 10: VTAB 20: INVERSE : PRINT
     " HIT 'R' TO RESTART ": NORMAL
460  POKE 768,250: POKE 769,2: CALL 770
470  GET A$: IF A$ = "R" THEN  RUN
480  END
490  REM  **MUSIC ML DATA**
500  DATA  172,01,03,174,01,03,169,04,3
     2,168,252,173,48,192,232,208,253,1
     36,208,239,206,0,03,208,231,96
510  REM  **ASCII DATA FOR KEYBOARD**
520  DATA  49,50,51,52,53,54,55,56,57,4
     8,58,45,0
530  DATA  81,87,69,82,84,89,85,73,79,8
     0,0
540  DATA  65,83,68,70,71,72,74,75,76,5
     9,0
```

```
550  DATA  32,90,88,67,86,66,78,77,44,4
     6,47,-1
```

Program 3: VIC Version

```
100 PRINT"{CLEAR}{03 DOWN}{04 RIGHT}{REV}T
    YPING{OFF} {REV}TEACHER{OFF}{06 DOWN}"

110 CCNT=0:POKE 36878,10:X=RND(-TI):REM ZE
    RO CHAR COUNTER AND SET VOLUME
120 S2=36875:S4=36877:REM SPEAKER NUMBERS
130 REM *ROUTINE TO ENTER CHARACTER POSITI
    ON DATA*
140 FOR ROW=1 TO 4:REM ROW DATA TO POKE
150 PRINT"{05 RIGHT}";
160 FOR COL=1 TO 12:REM COLUMN DATA TO POK
    E
170 READ CHAR
180 IF CHAR=0 THEN NEXT ROW:GOTO 160
190 IF CHAR=-1 THEN 260
200 POKE 828+CHAR*2,COL:POKE 829+CHAR*2,RO
    W:REM POKE DATA POSITION
210 IF CHAR=32 THEN PRINT" ";:GOTO 230
220 PRINT"{REV}";CHR$(CHAR);
230 NEXT COL
240 PRINT"{DOWN}"
250 GOTO 160
260 PRINT"{OFF} "
270 REM **ZERO TIMER AND SELECT RANDOM CHA
    R**
280 PRINT"{03 DOWN}{RIGHT}{REV}HIT ANY KEY
     TO START{OFF}"
290 GET A$:IF A$="" THEN 290
300 PRINT"{UP}";:FOR I=1 TO 21:PRINT" ";:N
    EXT I
310 TI$="000000"
320 N=INT((RND(1)*49)+42):REM CHOOSE A RAN
    DOM CHARACTER
330 IF N=60 OR N=62 OR N=63 OR N=OLDCHAR T
    HEN 320
340 OLDCHAR=N
350 PRINT"{HOME}{06 DOWN}";SPC(10);CHR$(N)
360 IF TI>3600 THEN 580:REM TIME UP?
370 REM **PROCESS YOUR RESPONSE**
380 GET A$:IF A$="" THEN 360
390 REM*PLEASANT SOUND*
400 CHAR=ASC(A$):POKE S2,225:FOR I=1 TO 5:
    NEXT I:POKE S2,0
410 CCNT=CCNT+1
420 IF CHAR<>N THEN 490
430 GOSUB 520
440 PRINT CHR$(CHAR)
450 FOR I=1 TO 10:NEXT I
460 GOSUB 520:PRINT"{REV}";CHR$(CHAR);"{OF
    OFF}"
470 GOTO 320
480 REM *YOU ERRED*
490 CCNT=CCNT-1:POKE S4,130:FOR I=1 TO 10 ~
    :NEXT I:POKE S4,0
500 GOTO 320
510 REM*POSITION CURSOR OVER TYPED CHAR*
520 PRINT"{HOME}{09 DOWN}";
530 FOR I=2 TO PEEK(829+CHAR*2)*2:PRINT:NE
    XT I
540 PRINT"{04 RIGHT}";
550 FOR J=1 TO PEEK(828+CHAR*2):PRINT"{
    RIGHT}";:NEXT J
560 RETURN
570 REM**CALC AND PRINT SCORES**
580 PRINT"{HOME}{06 DOWN}{04 RIGHT}{REV}CH
    AR/MINUTE{OFF}";"=";CCNT
590 PRINT"{HOME}":FOR I=1 TO 18:PRINT:NEXT
     I:PRINT"{RIGHT}{REV} HIT 'R' TO ~
    RESTART {OFF}"
600 REM *SCORE SOUND*
610 FOR I=244 TO 252 STEP 2:POKE S2,I:FOR ~
    J=1 TO 50:NEXT J:NEXT I:POKE S2,0
620 GET A$:IF A$="" THEN 620
630 IF A$="R" THEN RUN
640 END
650 REM*ASCII DATA FOR KEYBOARD*
660 DATA 49,50,51,52,53,54,55,56,57,48,43,
    45,0
670 DATA 81,87,69,82,84,89,85,73,79,80,64,
    42,0
680 DATA 65,83,68,70,71,72,74,75,76,58,59,
    61,0
690 DATA 32,90,88,67,86,66,78,77,44,46,47,
    -1
```



"Typing Teacher," VIC-20 version.

Program 4: TI Version

```
100 DIM CHAR(23,30)
110 RANDOMIZE
120 D=20
130 F1=300
140 F2=4000
150 V1=10
160 V2=2
170 CALL CLEAR
180 FOR J=9 TO 12
190 CALL COLOR(J,2,14)
200 NEXT J
210 FOR J=2 TO 8
220 CALL COLOR(J,2,15)
230 NEXT J
240 IF R=82 THEN 270
250 RESTORE
260 CALL CLEAR
270 PRINT "{6 SPACES}typing teacher"
280 FOR I=1 TO 18
290 PRINT
300 NEXT I
310 REM  ZERO CHARACTER COUNTER AND
    TIME
320 CHARCNT=0
330 TIME=0
340 REM  ROUTINE TO ENTER CHARACTER
    POSITION DATA
350 FOR ROW=11 TO 23 STEP 3
```

```
360 FOR COL=6 TO 30 STEP 2
370 READ CHAR(ROW,COL)
380 IF CHAR(ROW,COL)=0 THEN 450
390 IF CHAR(ROW,COL)=-1 THEN 460
400 IF CHAR(ROW,COL)=32 THEN 430
410 CALL HCHAR(ROW,COL,CHAR(ROW,COL)
    )
420 GOTO 440
430 PRINT " ";
440 NEXT COL
450 NEXT ROW
460 PRINT
470 PRINT "  PRESS any key TO START"
    ;
480 CALL KEY(3,S,STATUS)
490 IF STATUS=0 THEN 480
500 CALL HCHAR(24,5,32,22)
510 REM  *CHOOSE A RANDOM NUMBER*
520 N=INT((RND*47)+44)
530 IF (N>=60)*(N<=64)+(N=45)+(N=58)
    +(N=OLDCHAR)THEN 520
540 OLDCHAR=N
550 CALL VCHAR(7,16,N)
560 REM  **PROCESS YOUR RESPONSE**
570 TIME=TIME+1
580 IF TIME>900 THEN 670
590 CALL KEY(0,CR,STATUS)
600 IF STATUS=0 THEN 570
610 CALL SOUND(D,F1,V1)
620 CHARCNT=CHARCNT+1
630 REM  ADD ONE TO TOTAL
640 TIME=TIME+12
650 GOTO 760
670 PRINT TAB(4);
680 PRINT "characters/minute= ";CHAR
    CNT
690 PRINT
700 PRINT "(5 SPACES)HIT r TO RESTAR
    T";
710 CALL KEY(3,R,STATUS)
720 IF STATUS=0 THEN 710
730 IF R=ASC("R")THEN 250
750 END
760 IF CR<>N THEN 860
770 FOR ROW=11 TO 23 STEP 3
780 FOR COL=6 TO 30 STEP 2
790 IF CHAR(ROW,COL)=N THEN 820
800 NEXT COL
810 NEXT ROW
820 CALL HCHAR(ROW-1,COL,N)
830 CALL HCHAR(ROW-1,COL,32)
840 CALL HCHAR(ROW-1,COL,N)
850 GOTO 520
860 CHARCNT=CHARCNT-1
870 CALL SOUND(D,F2,V2)
880 GOTO 520
890 REM  *ASCII DATA FOR KEYBOARD*
900 DATA 49,50,51,52,53,54,55,56,57,
    48,61,0
910 DATA 81,87,69,82,84,89,85,73,79,
    80,47,0
920 DATA 65,83,68,70,71,72,74,75,76,
    59,0
930 DATA 32,90,88,67,86,66,78,77,44,
    46,-1
```

©

TI-99 Match-Em

C. Regena

In addition to its primary purpose of captivating youngsters, this program also serves as a guide and example of how to create educational games on any subject.

This simple matching game is designed for young children. A screen of 16 squares is shown. Press the letters on two of the squares to try to match the shapes. If you "Match-Em," the shape will be drawn at the right side of the screen, and you won't be able to use those squares again (the shape is replaced by diagonal lines). There are eight pairs of shapes to try to match.

If you wish to stop the game at any time, press "S" and the placement of all the shapes will be shown. After each game you have the option of trying again – with the shapes scrambled in a different random order.

Other Applications

Take a look at the BASIC logic in this game, then design your own. You may wish to use the capabilities of the TI-99/4A graphics and draw other pictures – animals, people, designs, etc. Each shape here is drawn in a separate character set, and a random foreground and background color combination is chosen. Keep your drawing to eight or fewer graphics characters; you may also want to specify a certain foreground and background color.

You can make this matching game into an educational game. Instead of matching shapes, match an answer to a mathematics problem; match a capital city to its state; match a date to a historical event; match parts of a compound word. Whatever you want.

Programming Techniques

DIMensioned arrays start with a subscript of zero unless you specify OPTION BASE 1, which starts subscripts at 1. I used dimensioned numbers to keep track of the eight shapes (16 total) and various coordinates needed for graphics.

MX() and MY() are the X and Y coordinates to draw a shape at the right of the screen after it has been successfully matched. The coordinates depend on how many matches have been made.

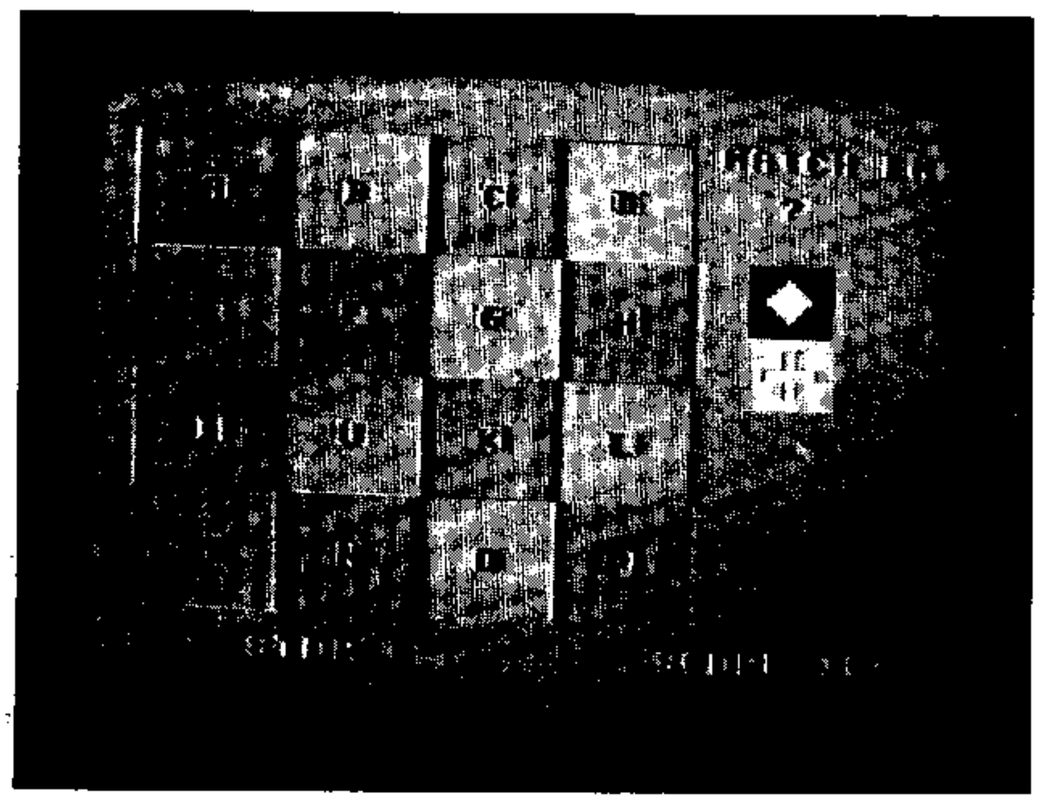
C1() and C2() are the X and Y coordinates for each square in the 16-square screen. D() indicates a red or a blue square.

A() and B() keep track of what shape is in which square. I use two arrays so that one can be a working array. B() also keeps track of the original order of the shapes when all the shapes are drawn (if you press "S" or if you have made all eight matches).

Lines 270-370 define graphics characters while the title screen is shown. Line 280 beeps a random sound for each character as it is defined. Graphics characters are defined by a string number. A null string is indicated either by "" or by two commas together and will yield a blank square for that graphics character. You do not need quote marks around the graphics string if it is in a data statement.

Lines 410-420 redefine the parentheses as a blue square and a red square. The game screen is then printed with lines 500-540. PRINTing characters is often faster than using the CALL HCHAR or CALL VCHAR method.

The shapes are numbered from 1 through 8. Lines 550-580 put the shape numbers in the B() array. Lines 600-660 mix up the members of the B() array and place them in the A() array. After a



Can you match the hidden symbols behind the colored squares?

B() is chosen for the A() array, it is set to zero so it won't be chosen again. Lines 670 to 690 set the B() array equal to the A() array so that the shapes can all be printed in the original order at the end of a game.

Lines 710-760 choose a random foreground color and a random background color for each shape, making sure that the foreground color is not the same as the background color.

Lines 1490-1590 are a subroutine to draw the shape starting at coordinates X and Y. CH is the character number and is calculated in line 1480, depending on the shape number.

Explanation Of The Program

Line Nos.

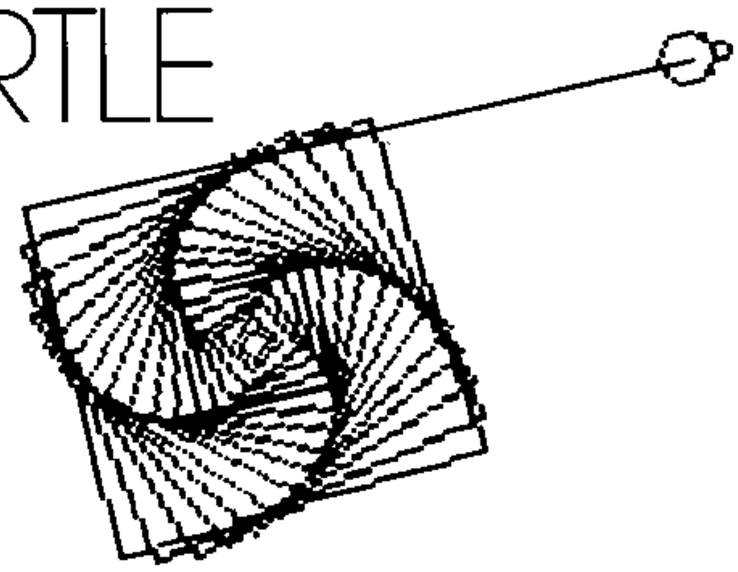
110-120	DIMension variables starting with a subscript of 1.
130-160	Read X and Y coordinates for matched shapes.
170-210	Read X and Y coordinates and character numbers for each of the 16 squares.
220-240	Print title screen.
250-260	Define functions for random variables used later.
270-370	Define graphics characters for character numbers 96 through 159 (eight shapes, each in a different character set).
380-400	Print instructions.
410-420	Define characters for red and blue squares.
430-450	Wait for player to press any key.
460-480	Clear screen and initialize score (number of tries) and number of matches.
490-540	Define colors and draw game screen.
550-580	Define B() elements as shape numbers 1 through 8 (two of each number).
590	Prints another line of game screen.
600-660	Randomly choose the order of the shapes in the 16 squares.
670-690	Set B() array elements equal to A() array.
700-760	Randomly choose colors for shapes.
770-790	Print name of game on screen.
800-840	Increment and print score.
850-900	Beep and wait for player to press a letter for first square.
910-990	Determine coordinates and draw diagonal lines if square has already been matched.
1000	Draws shape.
1010-1060	Beep and wait for player to press a letter for second square.
1070-1150	Determine coordinates and draw diagonal lines if square has already been matched.
1160	Draws shape.
1170-1220	Determine if a match has been made; if not, sounds "uh-oh."
1230-1300	If match has been made, these lines play arpeggio and determine coordinates, then draw shape at right of screen.
1310-1320	Set A() elements to zero so they cannot be used again for a correct match.
1330-1470	Cover squares again with red or blue square and return to next set of choices.
1480-1590	Subroutine to draw shape.
1600-1650	After all eight matches have been made, these lines play a tune.
1660-1710	Clear choices made and show all shapes on game screen.
1720-1750	Print option to play again, wait for player's choice, and branch appropriately.
1760-1770	Clear screen and end.

```
100 REM{3 SPACES}MATCH-EM
110 OPTION BASE 1
120 DIM A(16),B(16),C1(16),C2(16),D(1
    6),MX(8),MY(8)
130 FOR C=1 TO 8
140 READ MX(C),MY(C)
150 NEXT C
160 DATA 7,26,10,26,13,26,16,26,7,29,
    10,29,13,29,16,29
170 FOR C=1 TO 16
180 READ C1(C),C2(C),D(C)
190 NEXT C
200 DATA 3,5,40,3,10,41,3,15,40,3,20,
    41,8,5,41,8,10,40,8,15,41,8,20,40
210 DATA 13,5,40,13,10,41,13,15,40,13
    ,20,41,18,5,41,18,10,40,18,15,41,
    18,20,40
220 CALL CLEAR
230 CALL CHAR(64,"3C4299A1A199423C")
240 PRINT TAB(10);"MATCH-EM"::::::::
    :
250 DEF R=INT(RND*200+900)
260 DEF R15=INT(RND*15)+2
270 FOR C=96 TO 159
280 CALL SOUND(50,R,4)
290 READ C$
300 CALL CHAR(C,C$)
310 NEXT C
320 DATA "",,FFFFFFFFFFFFFFFF,,,,,,00
    000000000003CFF,010103030303010 1,F
    FFFFFFFFFFFFFFF,8080C0C0C0C0808,,
    FF3C,,""
330 DATA 0000000008081C1C,00000000000
    00101,3E3E7F7FFFFFFFFF,0000000080
    80C0C,03030707,FFFFFFFF,E0E0F0F,""
340 DATA "",0F0F0F0F0F0F0F0F,FFFFFFFF
    FFFFFFFF,F0F0F0F0F0F0F0F,,,,,0000
    0000010387CFE,0103070F070301
350 DATA FFFFFFFFFFFFFFFE,0080C0E0C08
    ,,7C381,,,0000000010103 83C,001F07
    0100010103,7CFFFFFFFEFFEFC7
360 DATA 00F0C0000000008,030706,8301,
    80C0C,,0000000000003C7E,000103030
    30301,FFE7C38181C3E7FF,0080C0C0C0
    C08,""
370 DATA 7E3C,,,000000003C3C3C3C,0000
    0F0F0F0F,3C3CFFFFFFFF3C3C,0000F0F
    0F0F,,3C3C3C3C,,""
380 CALL CLEAR
390 PRINT "PRESS TWO LETTERS.":: "TRY
    TO MATCH THE SHAPES.":: "THE BETTE
    R YOU ARE,THE"
400 PRINT :"LOWER YOUR SCORE WILL BE.
    ":::"PRESS 'S' TO STOP THE GAME":
    :"AND SEE
ALL THE SHAPES."
410 CALL CHAR(40,"FFFFFFFFFFFFFFFF")
420 CALL CHAR(41,"0")
430 PRINT :::"PRESS ANY KEY TO START.
    ";
440 CALL KEY(0,K,S)
450 IF S<1 THEN 440
460 CALL CLEAR
470 SC=0
480 M=0
490 CALL COLOR(2,5,9)
500 PRINT "((((()))))((((()))))":"(((
    ()))))((((()))))":"((A(())B))((C
    ()))D))":"((((()))))((((()))))"
510 PRINT "((((()))))((((()))))":")))
    ))((((()))))((((":")))))((((()))
    ))((((":"))E))((F(())G))((H((" 
```

```
520 PRINT "))))))((((()))))))((((":")))
    ))((((()))))))((((":"((((()))))))(((
    (()))))))":"((((()))))))((((()))))))"
530 PRINT "((I(())J))((K(())L))":"(((
    (()))))))((((()))))))":"((((()))))))(((
    (()))))))":"))))))((((()))))))((((("
540 PRINT "))))))((((()))))))((((":"))M
    ))((N(())O))((P(":")))))((((()))))
    ))((((":")))))((((()))))))((((("
550 FOR C=1 TO 8
560 B(C)=C
570 B(C+8)=C
580 NEXT C
590 PRINT :"S = STOP";TAB(20);"SCORE
    ="
600 FOR C=1 TO 16
610 RANDOMIZE
620 RC=INT(16*RND)+1
630 IF B(RC)=0 THEN 620
640 A(C)=B(RC)
650 B(RC)=0
660 NEXT C
670 FOR C=1 TO 16
680 B(C)=A(C)
690 NEXT C
700 M=0
710 FOR C=1 TO 8
720 F(C)=R15
730 F2(C)=R15
740 IF F2(C)=F(C)THEN 730
750 CALL COLOR(C+8,F(C),F2(C))
760 NEXT C
770 FOR C=1 TO 8
780 CALL HCHAR(2,23+C,ASC(SEG$("MATCH
     EM",C,1)))
790 NEXT C
800 SC=SC+1
810 S$=STR$(SC)
820 FOR C=1 TO LEN(S$)
830 CALL HCHAR(23,27+C,ASC(SEG$(S$,C,
    1)))
840 NEXT C
850 CALL SOUND(150,1397,2)
860 CALL HCHAR(4,26,63)
870 CALL KEY(0,K,S)
880 IF K=83 THEN 1660
890 IF (K<65)+(K>80)THEN 870
900 CALL HCHAR(4,26,K)
910 N=K-64
920 A1=N
930 X=C1(N)
940 Y=C2(N)
950 IF A(N)<>0 THEN 1000
960 CALL HCHAR(X,Y-1,92,3)
970 CALL HCHAR(X+1,Y-1,92,3)
980 CALL HCHAR(X+2,Y-1,92,3)
990 GOTO 1010
1000 GOSUB 1480
1010 CALL SOUND(150,1397,2)
1020 CALL HCHAR(4,29,63)
1030 CALL KEY(0,K,S)
1040 IF K=83 THEN 1660
1050 IF (K<65)+(K>80)THEN 1030
1060 CALL HCHAR(4,29,K)
1070 N=K-64
1080 A2=N
1090 X=C1(N)
1100 Y=C2(N)
1110 IF A(N)<>0 THEN 1160
1120 CALL HCHAR(X,Y-1,92,3)
1130 CALL HCHAR(X+1,Y-1,92,3)
1140 CALL HCHAR(X+2,Y-1,92,3)
1150 GOTO 1170
1160 GOSUB 1480
1170 IF A(A1)=0 THEN 1200
1180 IF A(A2)=0 THEN 1200
1190 IF A(A1)=A(A2)THEN 1230
1200 CALL SOUND(150,330,2)
1210 CALL SOUND(150,262,2)
1220 GOTO 1340
1230 M=M+1
1240 X=MX(M)
1250 Y=MY(M)
1260 CALL SOUND(150,262,2)
1270 CALL SOUND(150,330,2)
1280 CALL SOUND(150,392,2)
1290 CALL SOUND(300,523,2)
1300 GOSUB 1500
1310 A(A1)=0
1320 A(A2)=0
1330 IF M=8 THEN 1600
1340 X=C1(A2)
1350 Y=C2(A2)
1360 CALL HCHAR(X,Y-1,D(N),3)
1370 CALL HCHAR(X+1,Y-1,D(N),3)
1380 CALL HCHAR(X+2,Y-1,D(N),3)
1390 CALL HCHAR(X+1,Y,N+64)
1400 X=C1(A1)
1410 Y=C2(A1)
1420 CALL HCHAR(X,Y-1,D(A1),3)
1430 CALL HCHAR(X+1,Y-1,D(A1),3)
1440 CALL HCHAR(X+2,Y-1,D(A1),3)
1450 CALL HCHAR(X+1,Y,A1+64)
1460 CALL HCHAR(4,26,32,4)
1470 GOTO 800
1480 CH=8*(B(N)-1)+96
1490 CALL SOUND(150,-1,2)
1500 CALL HCHAR(X,Y-1,CH+7)
1510 CALL HCHAR(X,Y,CH)
1520 CALL HCHAR(X,Y+1,CH+7)
1530 CALL HCHAR(X+1,Y-1,CH+1)
1540 CALL HCHAR(X+1,Y,CH+2)
1550 CALL HCHAR(X+1,Y+1,CH+3)
1560 CALL HCHAR(X+2,Y-1,CH+4)
1570 CALL HCHAR(X+2,Y,CH+5)
1580 CALL HCHAR(X+2,Y+1,CH+6)
1590 RETURN
1600 RESTORE 1610
1610 DATA 262,330,392,523,330,392,523
     ,659,392,523,659,784,523,659,784
     ,1046,1046
1620 FOR C=1 TO 17
1630 READ J
1640 CALL SOUND(-99,J,2)
1650 NEXT C
1660 CALL HCHAR(4,26,32,4)
1670 FOR N=1 TO 16
1680 X=C1(N)
1690 Y=C2(N)
1700 GOSUB 1480
1710 NEXT N
1720 PRINT :"PLAY AGAIN? [Y N]";
1730 CALL KEY(0,K,S)
1740 IF K=78 THEN 1760
1750 IF K=89 THEN 460 ELSE 1730
1760 CALL CLEAR
1770 END
```

©

FRIENDS OF THE TURTLE



David D Thornburg, Associate Editor

The Readers Write

One of the greatest pleasures I have in writing these columns comes when the readers teach me something new. Sometimes, I say something that isn't quite true, and a reader thoughtfully brings the correction to my attention. One recent example of this is the topic of recursion and Atari PILOT.

I have stated that one cannot write recursive programs in PILOT because PILOT doesn't have local variables. If you have read the columns on recursion that appeared a few months ago, you may have been impressed with the compactness of some of the Logo procedures that take advantage of recursion.

COMPUTE! reader Aaron Cohen is an avid Atari PILOT enthusiast who has found a way to write recursive programs in PILOT so that he can create fractal patterns and other self-referenced curves without a lot of typing. As he points out, the problem isn't overwhelming. Since Atari PILOT allows a procedure to use itself (to a maximum of eight times), the only thing preventing true recursion is parameter passing and keeping track of the levels. His solution to this latter problem is deceptively simple. He calculates a variable #L to the desired depth of the recursion, and decreases this level each time he goes into the procedure. Each time he leaves the procedure, he increases the value of #L. In between, you do everything much as you would in Logo.

To see how this works, look at the program listing for a binary tree, *TREE. In line 30 we set #L equal to 64. The procedure *BRANCH starts out by setting #L to one-half its previous value. It then draws a line of length #L (which is now 32), and turns to the left by 45 degrees. Next, *BRANCH is used again, since #L is not equal to 1. This process is repeated until #L equals 1, at which point the turtle draws the other branch of the smallest twig, and repeats this process for all the other branches. In this procedure, the value in #L is used both as a level counter and as the length of the drawn line.

The next program provided by reader Cohen draws a Hilbert curve, and is a PILOT adaptation of a Logo program that appeared in Abelson and diSessa's *Turtle Geometry*. When entering this program, you can take advantage of the Atari screen editor in the following way. Enter the program from the AUTO mode through line 200. List the program and then move the cursor to line 40. By retyping the new line numbers (for lines 210 through 370) and editing the slight differences, you can save a lot of time and minimize your chances for typing errors. The Hilbert curve is one of those mathematical curiosities that fills a plane when the step size is reduced to zero. The level drawn by Aaron's program is quite attractive.

Finally, being a student at the University of Michigan, Aaron couldn't resist sending me his maize and blue "Big M" fractal based on the shape of a block letter M. As you can see from the listing, this is probably the easiest of the PILOT recursive programs to understand.

Now who said that Atari Pilot was just a kiddies' language?

The National Logo Exchange

In the interest of keeping **COMPUTE!**'s readers as fully informed as possible, all Friends of the Turtle should know about the National Logo Exchange. This group in Charlottesville, Virginia, publishes a noncommercial newsletter monthly from September through May (subscription $25). I have looked at a few copies of their newsletter and find it to contain material of special interest to teachers, as well as being a source of interesting programming ideas in general. We try to be as informative as possible, but the true Logophile will want to also keep up to date with the newsletters from the Young People's Logo Association (1208 Hillsdale Dr., Richardson, TX 75081) and the National Logo Exchange (P.O. Box 5341, Charlottesville, VA 22905).

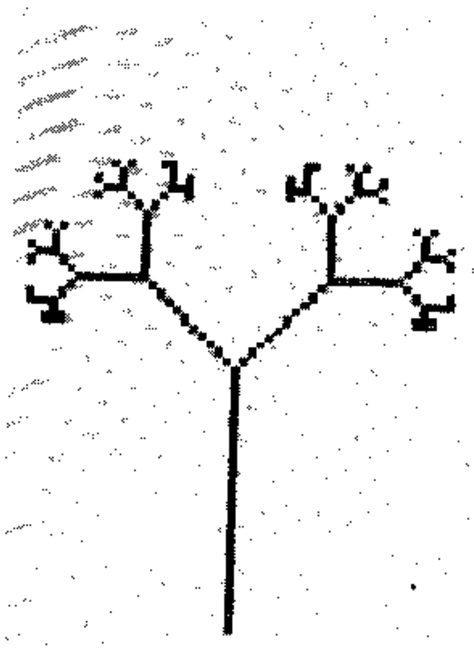
Speaking Of YPLA

I recently received a copy of an excellent book – the *Turtle's Sourcebook* – from the YPLA (address above). This sourcebook is perfect for anyone who teaches turtle graphics or Logo to children.

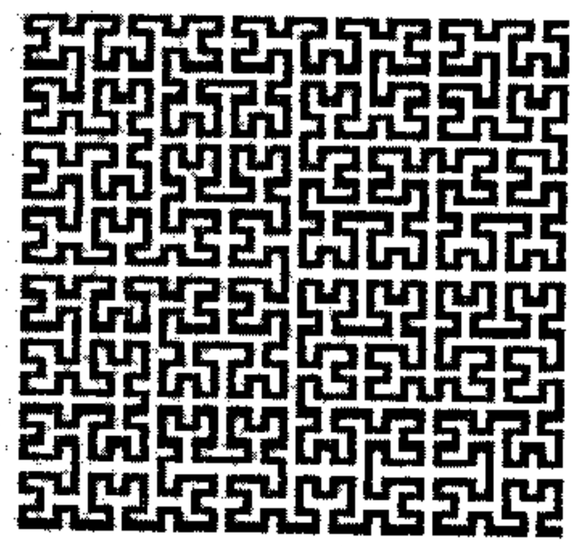
The authors, Jim Muller and Donna Bearden of YPLA, and Kathleen Martin at the University of Dallas, have done an excellent job compiling reference material, projects, worksheets, and general programming material. If you teach programming, and turtle graphics in particular, you will find the *Turtle's Sourcebook* to be of great value.

Next Time

The robots are coming, the robots are coming....



Tree



Hilbert Curve

Michigan

Program 1.

```
10 *TREE
20 GR:CLEAR; GOTO 0,-30;TURNTO 0; PEN BLUE
30 C:#L=2*2*2*2*2*2
40 *BRANCH
50 C:#L=#L/2
60 GR:DRAW #L
70 GR:TURN -45
80 U(#L<>1):*BRANCH
90 GR:TURN 90
100 U(#L<>1):*BRANCH
110 GR:TURN -45; DRAW -#L
120 C:#L=#L*2
130 E:
```

Program 2.

```
10 *HILBERT
20 GR:CLEAR; GOTO 30,-20; TURNTO 0
30 C:#L=6
40 *LHILBERT
50 C:#L=#L-1
60 J(#L=0):*LEND
70 GR:TURN -90
80 U:*RHILBERT
90 GR:DRAW 2
100 GR:TURN 90
110 U:*LHILBERT
120 GR:DRAW 2
130 U:*LHILBERT
140 GR:TURN 90
150 GR:DRAW 2
160 U:*RHILBERT
170 GR:TURN -90
180 *LEND
190 C:#L=#L+1
200 E:
210 *RHILBERT
220 C:#L=#L-1
230 J(#L=0):*REND
240 GR:TURN 90
250 U:*LHILBERT
260 GR:DRAW 2
270 GR:TURN -90
280 U:*RHILBERT
290 GR:DRAW 2
300 U:*RHILBERT
310 GR:TURN -90
320 GR:DRAW 2
330 U:*LHILBERT
340 GR:TURN 90
350 *REND
360 C:#L=#L+1
370 E:
```

Program 3.

```
10 *MICHIGAN
20 GR:CLEAR; PEN YELLOW; GOTO -60,-10; TU
   RNTO 90
30 C:@B710=7*16
40 C:@B712=7*16
50 C:#L=4
60 *UOFM
70 C:#L=#L-1
80 GR(#L=0):DRAW 2
90 GR:TURN -90
100 U(#L<>0):*UOFM
110 GR:TURN 90
120 U(#L<>0):*UOFM
130 GR:TURN 60
140 U(#L<>0):*UOFM
150 GR:TURN -120
160 U(#L<>0):*UOFM
170 GR:TURN 60
180 U(#L<>0):*UOFM
190 GR:TURN 90
200 U(#L<>0):*UOFM
210 GR:TURN -90
220 U(#L<>0):*UOFM
230 C:#L=#L+1
240 E:
```

©

C. Regena

Secondary Education

One of the early complaints about the TI was the lack of educational software for the secondary school level (junior high, middle, or senior high schools). The Scott, Foresman company developed excellent courseware in mathematics and reading for the elementary grades (starting with the primary grades for their first modules). Many users wondered if their children would "outgrow" the computer. Is the TI only for younger children?

The answer is that the powerful graphics and sound capabilities make the TI an excellent learning tool for young children, but there is no reason we cannot use the same computer for older children (and for adults with home and business applications).

In the last year the software growth rate has been phenomenal, including "third party" educational software for the TI. The computer can be used in just about any subject area. New software companies and new products are being created daily. I'm going to review a few applications for older students here; but keep in mind that even between the time I write this column and the time it is published, many more products will probably be announced.

Educational Modules

Texas Instruments has several modules that could be used in the junior high, middle school, or senior high school. Weight and Nutrition is a module that could be used by secondary students studying health or home economics.

Music students (and even non-musicians) can compose with the Music Maker command module. There are several options, including one in which short lines are placed on the screen and moved up or down as desired. Press a key and listen to the pattern you just created. One of the options lets you choose notes and rests and place them on a staff. You may choose a key signature and time signature. As you place the notes on the staff you can see, for example, what proportion of the measure a quarter note requires. When you finish the measure, you may listen to it or go to the next measure. You may play more than one note at a time if you wish. And if you compose something really special, you can then save your masterpiece on cassette.

Music students will also enjoy programming their own music either to learn a difficult piece, to sing along with, or to use as accompaniment for a solo instrument. You don't often think of using a computer in a music class, but because of the excellent sound capabilities of the TI the music departments may soon be begging for their own computers.

The Home Financial Decisions module could be a boon to economics classes. No longer do you need to find the right table in the back of the textbook, pick the right formula, interpolate, etc. Use the TI computer and this module. Suppose I want to buy a house and need to borrow some money. Press 1 for loans, press 2 for size of payments. Enter $65,000 for the loan, 360 monthly payments, and perhaps an interest rate of 12.5. I can find out immediately that the monthly payment is $693.72.

For physics and engineering students Texas Instruments has disk or cassette software called Electrical Engineering Library and Structural Engineering Library. Texas Instruments also has a Math Routine Library for advanced math students. Many times those tough equations that used to take hours or days to solve may now be solved easily and quickly with the computer.

The TI-99/4A keyboard has the letters in the same positions as those on a standard typewriter, and the shapes of the keys are similar, so the computer is ideal for touch-typing students. Students may use the Texas Instruments Touch Typing command module.

The Addison-Wesley Publishing Company has Computer Math Games, and Scott, Foresman has Math Action Games for grades one through eight or nine. What a way to practice math skills – by playing a video game! Milliken Math is also developing a math drill and practice series for grades one through eight.

The Minnesota Educational Computing Consortium (MECC) is renowned for its educational software for grades one through eight in a variety of math and science subjects. Their software is being developed for the TI computer on diskette.

Control Data Publishing Company is another pioneer in computer-aided instruction with their PLATO programs for all ages in all subjects. The first programs available for the TI (also for Atari 800 and Apple II Plus) are math, physics, French, German, and Spanish. For the TI you need the 32K memory expansion, disk controller, one disk drive, and the PLATO interpreter cartridge.

Math Competency Programs

Below are two short programs for secondary school students. These are called "Math Competency" because these types of problems are found in SRA, ACT, or other high school standardized competency tests. Younger students (third grade and up) should also be able to use the programs.

"Buying Items" gives a list of five items with their prices. The first question requires a total cost for all five items. The second question asks which two items may be purchased with a given amount of money. The question is in multiple-choice form.

"Earning Money" is a program using hourly or weekly wages to find a total earned for a given amount of time.

If you enter incorrect answers, you will be reminded how to get the right answer, and you will be given the same type of problem again. If you enter correct answers, you have the choice of solving another of the same kind of problem or continuing on to different sorts of questions.

Programming Techniques

"Buying Items"

There are three different categories for price lists. The number A is chosen randomly to be 1, 2, or 3. School supplies is number 1, a toy store is number 2, and a grocery store is number 3. The items I$ are read in as data in an array I$(A,C), where I$(2,4) would mean the name of an item in category 2 (toys), the fourth item listed.

The data for each item includes a minimum price I(A,C,1) and a maximum price I(A,C,2). For the actual price list for the problem, the price P is a random number from the minimum to the maximum:

```
D=I(A,C,2)-I(A,C,1)
P=I(A,C,1)+INT(RND*D+1)
```

A subroutine is used to convert the price calculated as a number of cents to a dollar value for printing in the problem. The price P is a whole number of cents. For example, 9 would be 9 cents; 59 would be 59 cents; and 135 would be 135 cents. To get the computer to print a decimal number that may include zeros for dollars, I use string manipulation. First let P$ be the string value of P.

If the length of P$ is 1, that means there is a single digit. In dollars we'll need a leading zero, so P$="0"&P$. Next I check to see if we have only cents – a length of 2 – because if there are only cents I want a space between the dollar sign and the decimal point. Therefore, if LEN (P$) is equal to 2, then P$=" "&P$. Now I put the right two characters to the right of a decimal point, and whatever is to the left are dollars. The subroutine is:

```
460 P$=STR$(P)
470 IF LEN(P$)>1 THEN 490
480 P$="0"&P$
490 IF LEN(P$)>2 THEN 510
500 P$=" "&P$
510 PR$=SEG$(P$,LEN(P$)-1,2)
520 PL$=SEG$(P$,1,LEN(P$)-2)
530 P$="$"&PL$&"."&PR$
540 RETURN
```

To combine string variables, an ampersand sign is used rather than a plus. In TI BASIC, IF-THEN-ELSE statements must contain statement numbers rather than commands. STR$ changes a number to a string. LEN(P$) finds the length or the number of characters in P$. SEG$(P$,A,B) yields the segment of the string P$ starting with the character in spot number A and containing the number B characters.

"Earning Money"

The names of the people in the problems are read in as N$(I) and T$(I) where I is a subscript from 0 to 5. The ways of earning money are read in as phrases J$(I).

The wage earned is P=100+25*INT(RND*10), which will translate from a dollar to as high as $3.25, in amounts divisible by 25¢. With this program, the amount earned, P, is known to be at least $1, so the subroutine for printing the dollar amount is:

```
340 P$=STR$(P)
350 P$="$"&SEG$(P$,1,LEN(P$)-2)&"."&SEG$(P$,
    LEN(P$)-1,2)
360 RETURN
```

A name is chosen with the random number N, and the number of hours in the first problem is a random number H=8+INT(RND*11). For the second type of problem, the number of weeks is a random number W=INT(RND*19)+2, which can be from two weeks to 20 weeks. The third type of problem chooses a random name, a random job, and a random number of weeks W=INT(RND*8)+2, which is from two weeks to nine weeks.

Program 1.

```
100 CALL CLEAR
110 PRINT TAB(6);"MATH COMPETENCY"
120 CALL CHAR(136,"080402FF020408")
130 PRINT :::TAB(7);"BUYING ITEMS"
140 CALL COLOR(14,9,16)
```

```
150 PRINT :::::TAB(9);"BY REGENA":::
    ::
160 DIM I$(3,5),I(3,5,2),N$(6),J(5),
    H$(3),S$(4)
170 FOR C=1 TO 6
180 READ N$(C)
190 NEXT C
200 FOR A=1 TO 3
210 FOR C=1 TO 5
220 READ I$(A,C),I(A,C,1),I(A,C,2)
230 NEXT C
240 NEXT A
250 DATA ANGIE,CINDY,CHERY,RICKY,BOB
    BY,RANDY,PENCIL,8,15
260 DATA ERASER,2,10,NOTEBOOK,35,99,
    RULER,29,49
270 DATA PAPER,59,90,DOLL,249,599,BA
    LL,49,89,TRUCK,100,150
280 DATA GAME,270,500,MODEL,300,700,
    CANDY,20,50
290 DATA MEAT,123,425,FRUIT,24,50,CH
    IPS,100,257,BREAD,100,179
300 H$(1)="PENCIL AND ERASER"
310 H$(2)="BALL AND TRUCK"
320 H$(3)="CANDY AND FRUIT"
330 GOTO 550
340 PRINT TAB(15);"PRESS <ENTER>";
350 CALL KEY(0,K,S)
360 IF K<>13 THEN 350
370 RETURN
380 CALL SOUND(100,330,2)
390 CALL SOUND(150,262,2)
400 RETURN
410 CALL SOUND(100,262,2)
420 CALL SOUND(100,330,2)
430 CALL SOUND(100,392,2)
440 CALL SOUND(200,523,2)
450 RETURN
460 P$=STR$(P)
470 IF LEN(P$)>1 THEN 490
480 P$="0"&P$
490 IF LEN(P$)>2 THEN 510
500 P$=" "&P$
510 PR$=SEG$(P$,LEN(P$)-1,2)
520 PL$=SEG$(P$,1,LEN(P$)-2)
530 P$="$"&PL$&"."&PR$
540 RETURN
550 RANDOMIZE
560 A=INT(RND*3+1)
570 TP=0
580 CALL CLEAR
590 PRINT "GIVEN THIS PRICE LIST:"::
600 FOR C=1 TO 5
610 D=I(A,C,2)-I(A,C,1)
620 P=I(A,C,1)+INT(RND*D+1)
630 GOSUB 460
640 TP=TP+P
650 PRINT TAB(6);I$(A,C);TAB(15);P$
660 NEXT C
670 R=INT(RND*13+4)
680 CALL COLOR(13,R,R)
690 CALL HCHAR(18,6,128,18)
700 CALL VCHAR(19,6,128,5)
710 CALL VCHAR(19,23,128,5)
720 CALL HCHAR(24,6,128,18)
730 F=INT(RND*2+1)
740 IF F=2 THEN 790
750 PRINT ::"HOW MUCH WILL IT COST"
760 PRINT "TO BUY ALL THE ITEMS"
770 PRINT "ON THE LIST?"
780 GOTO 830
790 N=INT(RND*6+1)
800 PRINT ::N$(N);" WANTS TO BUY"
810 PRINT "EVERYTHING ON THE LIST."
820 PRINT "WHAT WILL THE TOTAL COST
    BE?"
830 INPUT "$":X
840 IF ABS(X-TP/100)<.001 THEN 920
850 GOSUB 380
860 PRINT :"ADD ALL FIVE NUMBERS."
870 P=TP
880 GOSUB 460
890 PRINT "THE TOTAL IS ";P$:::
900 GOSUB 340
910 GOTO 550
920 GOSUB 410
930 CALL HCHAR(20,1,32,128)
940 IF F=2 THEN 970
950 PRINT "IF YOU COULD ONLY SPEND"
960 GOTO 980
970 PRINT "IF ";N$(N);" COULD ONLY S
    PEND"
980 IF A<>1 THEN 1010
990 M=INT(RND*5+25)
1000 GOTO 1050
1010 IF A<>2 THEN 1040
1020 M=INT(RND*36+239)
1030 GOTO 1050
1040 M=INT(RND*18+100)
1050 P=M
1060 GOSUB 460
1070 PRINT P$;", WHICH OF THESE PAIR
     S"
1080 PRINT "OF ITEMS COULD ";
1090 IF F<>1 THEN 1120
1100 PRINT "YOU BUY?"::
1110 GOTO 1160
1120 IF N>3 THEN 1150
1130 PRINT "SHE BUY?"::
1140 GOTO 1160
1150 PRINT "HE BUY?"::
1160 R=INT(RND*4+1)
1170 FOR V=1 TO 4
1180 IF V=R THEN 1280
1190 X=INT(RND*2+4)
1200 S$(V)=I$(A,X)
1210 X=INT(RND*3+1)
1220 S$(V)=S$(V)&" AND "&I$(A,X)
1230 IF V=1 THEN 1290
1240 FOR V1=1 TO V-1
1250 IF S$(V1)=S$(V)THEN 1190
1260 NEXT V1
1270 GOTO 1290
1280 S$(V)=H$(A)
1290 PRINT TAB(3);CHR$(64+V);" "&S$(
     V)
1300 NEXT V
1310 CALL SOUND(150,1397,2)
1320 CALL KEY(0,K,S)
1330 IF (K<65)+(K>68)THEN 1320
1340 CALL HCHAR(K-45,4,42)
1350 IF K<>64+R THEN 1410
1360 GOSUB 410
1370 PRINT :"TRY AGAIN? (Y/N)";
1380 CALL KEY(0,K,S)
1390 IF K=89 THEN 550
1400 IF K=78 THEN 1450 ELSE 1380
1410 GOSUB 380
1420 CALL HCHAR(19+R,3,136)
1430 PRINT :"THE TOTAL OF THE TWO IT
     EMS  MUST BE LESS THAN ";P$
1440 GOTO 1370
1450 CALL CLEAR
1460 END
```

Program 2.

```
100 CALL CLEAR
110 PRINT TAB(6);"MATH COMPETENCY"
120 PRINT :::TAB(7);"EARNING MONEY"
130 PRINT :::::TAB(9);"BY REGENA":::
    ::
140 DIM N$(5),J$(5),T$(5)
150 FOR I=0 TO 5
160 READ N$(I),J$(I),T$(I)
170 NEXT I
180 DATA SAM,DOING ODD JOBS,JOHN,JOE
    ,MOWING LAWNS,ANDY,BOB,TENDING C
    HILDREN,MARK,ANN
190 DATA RUNNING ERRANDS,LENA,SUE,DO
    ING HOUSEWORK,AURA,KAY,DELIVERIN
    G ADS,DAWN
200 GOTO 370
210 PRINT :TAB(15);"PRESS <ENTER>";
220 CALL KEY(0,K,S)
230 IF K<>13 THEN 220
240 RETURN
250 CALL SOUND(100,330,2)
260 CALL SOUND(150,262,2)
270 RETURN
280 CALL SOUND(100,262,2)
290 CALL SOUND(100,330,2)
300 CALL SOUND(100,392,2)
310 CALL SOUND(200,523,2)
320 RETURN
330 P=100+25*INT(RND*10)
340 P$=STR$(P)
350 P$="$"&SEG$(P$,1,LEN(P$)-2)&"."&
    SEG$(P$,LEN(P$)-1,2)
360 RETURN
370 CALL CLEAR
380 RANDOMIZE
390 N=INT(RND*6)
400 H=8+INT(RND*11)
410 GOSUB 330
420 PRINT N$(N);" WORKS";H;"HOURS PE
    R WEEK."
430 IF N<3 THEN 460
440 PRINT :"SHE EARNS ";
450 GOTO 470
460 PRINT :"HE EARNS ";
470 PRINT P$;" PER HOUR."
480 IF N<3 THEN 510
490 PRINT :"HOW MUCH DOES SHE EARN"
500 GOTO 520
510 PRINT :"HOW MUCH DOES HE EARN"
520 PRINT :"IN A WEEK?"::
530 INPUT "$":D
540 D1=P*H/100
550 IF ABS(D-D1)>.001 THEN 610
560 GOSUB 280
570 PRINT ::"TRY AGAIN? (Y/N)"
580 CALL KEY(0,K,S)
590 IF K=89 THEN 370
600 IF K=78 THEN 680 ELSE 580
610 GOSUB 250
620 PRINT :"MULTIPLY";H;"HOURS BY ";
    P$::"PER HOUR."
630 P=H*P
640 GOSUB 340
650 PRINT :"THE ANSWER IS ";P$
660 GOSUB 210
670 GOTO 370
680 CALL CLEAR
690 RANDOMIZE
700 N=INT(RND*6)
710 H=INT(RND*11)+8
720 GOSUB 330
730 PRINT N$(N);" EARNS ";P$;" PER H
    OUR."
740 IF N<3 THEN 770
750 PRINT :"SHE WORKS";
760 GOTO 780
770 PRINT :"HE WORKS";
780 PRINT H;"HOURS PER WEEK."
790 IF N<3 THEN 820
800 PRINT :"HOW MUCH WILL SHE EARN I
    N"
810 GOTO 830
820 PRINT :"HOW MUCH WILL HE EARN IN
    "
830 W=INT(RND*19)+2
840 PRINT :W;"WEEKS?"::
850 INPUT "$":D
860 D1=P*H*W/100
870 IF ABS(D-D1)>.001 THEN 930
880 GOSUB 280
890 PRINT ::"TRY AGAIN? (Y/N)"
900 CALL KEY(0,K,S)
910 IF K=89 THEN 680
920 IF K=78 THEN 1030 ELSE 900
930 GOSUB 250
940 PRINT :"MULTIPLY";H;"HOURS BY"
950 PRINT :P$;" PER HOUR."
960 PRINT :"THEN MULTIPLY BY";W;"WEE
    KS."
970 PRINT :"THE ANSWER IS ";
980 P=H*P*W
990 GOSUB 340
1000 PRINT P$::
1010 GOSUB 210
1020 GOTO 680
1030 CALL CLEAR
1040 J=INT(RND*6)
1050 T=INT(RND*6)
1060 GOSUB 330
1070 W=INT(RND*8)+2
1080 PRINT T$(T);" EARNED ";P$;" LAS
     T WEEK"
1090 PRINT :J$(J);"."
1100 IF T<3 THEN 1130
1110 PRINT :"IF SHE EARNED THIS AMOU
     NT"
1120 GOTO 1140
1130 PRINT :"IF HE EARNED THIS AMOUN
     T"
1140 PRINT :"EVERY WEEK, WHAT WOULD
     THE"
1150 PRINT :"TOTAL INCOME BE FOR"
1160 PRINT :W;"WEEKS?"::
1170 INPUT "$":D
1180 D1=P*W/100
1190 IF ABS(D-D1)>.001 THEN 1250
1200 GOSUB 280
1210 PRINT ::"TRY AGAIN? (Y/N)";
1220 CALL KEY(0,K,S)
1230 IF K=89 THEN 1030
1240 IF K=78 THEN 1330 ELSE 1220
1250 GOSUB 250
1260 PRINT :"MULTIPLY ";P$;" PER WEE
     K"
1270 PRINT :"BY";W;"WEEKS."
1280 P=P*W
1290 GOSUB 340
1300 PRINT :"THE ANSWER IS ";P$::
1310 GOSUB 210
1320 GOTO 1030
1330 CALL CLEAR
1340 END
```

©

Estimating TI-99 Memory

Michael A. Covington

You know the feeling – you're in the last stages of typing in some tremendously complex and subtle BASIC program that has been your brainchild for the last two or three months, you decide to run part of it for testing, and you get the dreaded message

*** MEMORY FULL**

telling you that your program is too big for the computer, and that all your work has been in vain.

At this point, you may be strongly tempted to go right to your dealer and buy another 32K of RAM. If you can afford to do so, more power to you; but many of us can't, at least not on a moment's notice. Alternatively, you may start going through your program and trimming it down. The first thing to do would be cut out the REM statements, especially if there are a lot of them; then make the PRINT statements less verbose, start combining short LET statements on one line (without the word LET, of course!) if your computer will let you, and so forth. But wouldn't it have been nice to know, earlier in the game, that you were running out of memory?

Some computers give you a command or pseudovariable that will tell you how much memory is free. Others, however, don't; you have to work blind. But I've developed a simple trick for finding out roughly how much memory is free even without a BASIC command for doing it.

The first thing you have to do is find out how large a numeric array your computer can accommodate. Try the one-line program:

```
1 DIM Q(5000)
```

If you can RUN this without getting a "memory full" message, try changing it to DIM Q(10000), and so on, until you hit the maximum. Alternatively, if you do get a "memory full" message, try reducing the size of the array until you don't. By this method I've found out that a 16K TI-99 will allow

```
1 DIM Q(1812)
```

as an upper limit.

But wait a minute, you say. On the TI-99, each element in a numeric array occupies eight bytes. The whole array therefore occupies 14,496 bytes, which equals just over 14K (remember that 1K = 1024 bytes). What happened to the rest of the 16K that the machine ostensibly has?

The answer is that a certain amount of memory – about 2K, it looks like – is occupied by essential control areas and by the DIM statement itself. This is why you can't use arrays to measure the size of memory exactly – but you can make quite useful approximate measurements.

What you do is simply add to your program a statement such as

```
1 DIM Q(1000)
```

(where Q is a variable not used in the program itself) and run the program; if you don't get a "memory full" message, then you have at least 1000 numeric storage locations left (equivalent to 8000 bytes on the TI-99). Similarly, a successful DIM Q(500) tells you that, on the TI-99, you have at least a quarter of the 16K RAM still available.

When you run the program, be sure to make it do a large variety of the things it will normally be used to do. Also, if you use string variables, do something to make them as long as they will ever normally be, since, on the TI-99, string variables are allocated dynamically (the amount of memory they occupy depends on their actual length).

Finally, *be sure* to remove the DIM statement after conducting the text; otherwise, your program will give you "memory full" messages later on and you won't know why.