A Turning Point For Atari?
Report From The Winter Consumer Electronics Show

COMPUTE!

$2.95
April
1986
Issue 71
Vol. 8, No.4
$3.75 Canada
02193
ISSN 0194-357X

The Leading Magazine Of Home, Educational, And Recreational Computing

New Technologies: The Converging Digital Universe

Why Everything From Computers & CD-ROMs To TVs & Satellites Are On A Collision Course

Adding Power To ST BASIC

Valuable Techniques For Atari Programmers

Apple Hi-Res Graphics

Mix, Flip, And Copy High-Resolution Screens

64 Screen Genie

A Magic Method For Character Graphics

AmigaDOS Batch Files

Take Control Of Your Amiga

Atari FontMaker

Design Your Own Custom Characters

COMPUTE!

APRIL 1986
VOLUME 8
NUMBER 4
ISSUE 71

NOTE: See page 125 before typing in programs.

AP Apple, **Mac** Macintosh, **AT** Atari, **ST**, Atari ST, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **128** Commodore 128, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **AM** Amiga. *General interest.

TOLL FREE Subscription Order Line
800-247-5470 (In IA 800-532-1272)

COMPUTE! Publications,Inc. abc
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., 825 7th Ave., New York, NY 10019 USA. Phone: (212) 265-8360. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. POSTMASTER: Send address changes to: **COMPUTE!** Magazine, P.O. Box 10955, Des Moines, IA 50950. Second class postage paid at Greensboro, NC 27403 and additional mailing offices. Entire contents copyright ©1986 by COMPUTE! Publications, Inc. All rights reserved, ISSN 0194-357X.

Pressing CONTROL and HELP returns a value of 145. The statement POKE 732,0 clears location 732, so you can check for subsequent keypresses.

Apple IIe/IIc Compatibility

I'm interested in buying an Apple IIc computer. Can it use IIe hardware and software?

Carlos Aguayo

The Apple IIc computer is basically an Apple IIe that has been redesigned to take up as little space as possible. To keep the IIc small, Apple left out the IIe's expansion slots (where additional hardware can be attached), but added a built-in 5¼-inch disk drive. They also put the most common IIc expansion hardware (80-column video display, an extra 64K of memory, and two serial input/output ports) on the main board of the IIc. In addition, the IIc has some features that weren't available when the IIe appeared: an advanced 65C02 microprocessor and a character set called Mousetext *which contains extra characters especially for Macintosh-style icon- and menu-based programs. The newest version of the IIe (called Enhanced IIe) does have these extra features; dealers can upgrade an older IIe at a small cost.*

The IIc can run almost all IIe programs, as long as no special hardware is required. For instance, some music programs can communicate with instruments through a MIDI (Musical Instrument Digital Interface) adapter. This adapter must connect to an expansion slot, which is possible only on a IIe. Other programs sometimes expect a parallel I/O interface to attach a printer. Since the IIc has only serial I/O, it can't run that type of modification. Although the IIc has no expansion slots, its peripherals (serial ports, disk drives, etc.) act like they are built into certain slots. Apple tried to select the most commonly used slot for each peripheral (printer in slot 1, disk drive in slot 6). However, not everyone puts everything in the same place, and some programs may demand an unconventional configuration. IIe owners can rearrange the cards in their slots to run such programs, but IIc owners don't have this option.

The serial ports on the IIc generate standard RS-232 signals which can be used to communicate with most modems from any manufacturer. Many of the most popular printers are also available with RS-232 interfaces. But the IIc does not have standard connectors for these ports. To save space on the back panel of the computer, DIN-type connectors are used instead; as a result, you'll need special cables (available from Apple dealers) to attach serial peripherals.

When it comes to expandability, the IIe is much more flexible than the IIc. Almost any kind of peripheral can be attached through one of its slots, including parallel I/O ports, MIDI interfaces, hard disk drives, coprocessors, huge RAM expansion cards, and a host of other devices. However, some third-party companies have begun modifying the IIc to put in extras like additional memory and Z80 processors (to run the CP/M operating system, a popular IIe add-on). It's still more difficult than expanding a IIe, but it can be done.

IBM PUT And GET

I own a TI-99/4A and an IBM PCjr. Lately, I've been trying to convert some programs from TI to IBM. I have only one problem: the PUT and GET graphic statements in the IBM system. I really don't understand them. Could you show me a way to make an image and move it?

Billy Mobley

First, be aware that IBM BASIC has two types of GET and PUT statements: one for graphics and another for random files. The syntax for each type is different, so be sure you're using the graphics type. GET grabs the screen image within a specified rectangle and stores a copy of it in an array. PUT does just the opposite, putting the image from an array back onto the screen.

Several important rules apply to PUT and GET. Before using either command, you must be in a graphics mode (SCREEN 1, for example); neither PUT nor GET works on a text screen. The array that you GET a shape into must be a one-dimensional numeric array dimensioned to the proper size. Finally, you must GET before you can PUT.

The most difficult task is deciding what size to dimension the array. If the array is too small, it can't hold the graphics image, and the program won't work. The simplest solution is to try a large size like DIM A(500). It won't hurt to dimension it larger than necessary, but this method wastes memory. Here's a more efficient formula that tells you the minimum required size for the array:

INT((4+INT((x*res+7)/8)*y)/prec)

In this formula, the variable x *represents the width of the image in pixels;* y *is the height of the image;* res *is 1 for high resolution and 2 for medium resolution; and* prec *is the precision of the array (2 for integer, 4 for single precision, and 8 for double precision).*

GET must be followed by the screen coordinates of two opposite corners of the rectangular image, and the name of the array. For example, GET (0,0)–(19,29),A grabs a 20 × 30 pixel image at the top-left corner of the screen and stores it in array A. (Of course, you must first have an image on the screen. This can be done with DRAW.) With a high-resolution screen and a single-precision array, the formula above gives 23, so the dimension statement would be DIM A(23).

PUT is followed by the coordinates of the location on the screen where the top-left corner of the image is placed, then the name of the array, and an optional parameter for special effects. Five special effects are available: PSET, PRESET, AND, OR, and XOR. If no special effect is specified, XOR is assumed.

PSET displays the image exactly as it appeared when GET was used. PRESET displays a negative image. AND displays only those parts of the image that overlap an image already on the screen. OR superimposes the image onto an image already on the screen. XOR is a combination of AND and PRESET, reversing only those parts of the image that overlap an image already on the screen. The best way to understand exactly what these special effects do is to try them yourself. Using our example, PUT (200,100),A,PSET displays the image stored in the A array in the center of the screen.

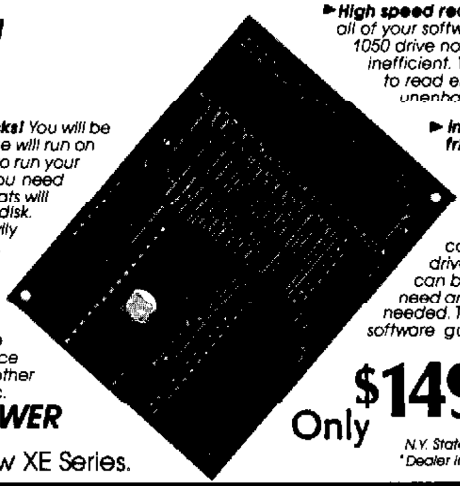
The operation of XOR may seem strange, but it's handy for animation. When you PUT using XOR twice in the same position, the screen is restored unchanged. This allows you to move an image over a background image, giving a 3-D effect. Animation with XOR is a three-step process: PUT the image on the screen with XOR, calculate the new position, PUT the image in the old position a second time to erase it. By performing these steps repeatedly, the image seems to move. The following program moves a ball across the screen.

```
1Ø SCREEN 1
2Ø DIM A(113)
3Ø CIRCLE (2Ø,2Ø),2Ø
4Ø PAINT (2Ø,2Ø)
5Ø GET (Ø,Ø)-(4Ø,4Ø),A
6Ø CLS
7Ø FOR C=1 TO 1ØØ
8Ø PUT (X1,Y1),A 'display ima
   ge
9Ø X2=X1+1:Y2=Y1+1 'calculate
    new position
1ØØ PUT (X1,Y1),A 'erase imag
    e
11Ø X1=X2:Y1=Y2 'old=new
12Ø NEXT 'repeat
```

Simpler Absent Printer Test

I'm writing with regard to the "Readers' Feedback" item on absent Commodore printers, published in the December 1985 COMPUTE!. Another way to avoid a DEVICE NOT PRESENT error is to access the appropriate device (4) through the command channel (15) and check the value of the status variable ST. If ST does not equal 0, then the printer is not present. Here is a short routine to demonstrate:

```
1Ø OPEN 15,4,15:CLOSE 15
2Ø IF ST<>Ø THEN 4Ø
```



```
30 OPEN 1,4:PRINT#1,"PRINTER I
      S ON":CLOSE 1:END
40 PRINT CHR$(147);"TURN ON PR
      INTER":GOTO 10
```

If you run this program with the printer off, it instructs you to turn the device on. Printing begins as soon as the printer is active.

Jim Plavecsky

Thanks for this compact, all-BASIC solution. In programs that open disk files or use an RS-232 device (usually a modem), you may want to perform this check at the very beginning, before you perform any other OPEN statements. The statement CLOSE 15 closes all other channels in addition to the command channel, terminating any RS-232 communications and disconnecting (but not really closing) any open disk files.

Emphasized TI Character Set

The custom character set given for the Commodore 64 on pages 108–109 of COMPUTE!'s January 1986 issue can be used on the TI-99/4A with only slight modifications. Since that character definition data is listed in hexadecimal format, it can be read as a *pattern-identifier* string and assigned with the CALL CHAR statement (see page II-76 in the *TI User's Reference Guide*). Each line in the Commodore character set listing contains data for a single character plus a checksum value at the end of the line. To convert the data in each line to a 16-character pattern-identifier string, type in the first eight two-digit hexadecimal numbers (spaces are left out, of course). In the first line, for instance, the resulting string could be used with a CALL CHAR statement to redefine the @ character. To create the new character set, first enter this program:

```
100 FOR L=1 TO 94
110 READ C$
120 CALL CHAR(L+32,C$)
130 PRINT CHR$(L+32);
140 NEXT L
150 GOTO 150
```

Next, you must enter a series of lines containing DATA statements. Each DATA statement represents the data for one character in the form of a 16-character pattern identifier string. For example, the first DATA line would look like this:

500 DATA 7CC6DEDEC0C07800

Here is how to enter all of the DATA lines.

lines	**500–800**	**Use data from line 7108–71F8 (defines ASCII characters 33–63)**
line	**810**	**Use data from line 7000 (ASCII 64)**
lines	**820–1070**	**Use data from lines 7208–72D0 (ASCII 65–90)**
line	**1080**	**Use data from line 70D8 (ASCII 91)**
line	**1090**	**Use 00C06030180C0600 as data (ASCII 92)**
lines	**1100–1110**	**Use data from lines 70E8–70F0 (ASCII 93–94)**
line	**1120**	**Use 00000000000000FF as data (ASCII 95)**
line	**1130**	**Use data from line 7200 (ASCII 96)**
lines	**1140–1390**	**Use data from lines 7008–70D0 (ASCII 97–122)**
lines	**1400–1420**	**Enter data from lines 72D8–72E8 (ASCII 123–125)**
line	**1430**	**Enter 000020745C080000 as data (ASCII 126)**

The result of your effort will be an emphasized font with true lowercase.

John Hedstrom

Thank you for your suggestion.

Tug-A-War

Mark Tuttle, Submissions Reviewer

Don't be fooled by the apparent simplicity of this two-player strategy game. It looks easy on the surface, but it's a stiff test of your concentration and ability to think ahead. The original version was written for the Commodore 64. We've added new versions for the Atari 400/800, XL, and XE, Apple II-series computers, Atari 520ST, Amiga, IBM PC/PCjr, and the TI-99/4A. Since the game is based on colors, every version requires a color monitor or TV. The IBM version requires BASICA and a color/graphics adapter for the PC or Cartridge BASIC for the PCjr. The Atari version requires at least 16K of RAM, and the Amiga version requires at least 512K.

Nearly everyone has played tug of war at one time or another. The traditional game pits two players or teams at opposite ends of a rope. At the middle of the rope is a flag, and each side tries to pull the flag into its territory. "Tug-a-War" is based on a similar concept. In this version, the flag is replaced with a round ball shape, and each player tries to maneuver the ball onto his or her side of the screen. Like many two-player games, the difficulty of Tug-a-War depends somewhat on the intelligence of your opponent. But even at the simplest level, you'll find that skill and foresight are essential to success.

Type in and save the appropriate program below. The rules are the same for every version (except Atari 520ST—see special instructions).

Battle Of The Colors

When you run Tug-a-War, two sets of colored boxes appear, one above the other. The lower, longer series of squares is the playing field. Near the middle of the playfield area is a round ball; the outermost boxes at each end of the playfield represent each player's home position. The players alternate turns, each trying to move the ball in their own direction, until it reaches one of the home squares.

So far, so good—but how do you move the ball? It's done not by pulling a rope, but by changing the colors of boxes in the playfield. The color of the square under the ball determines which direction it moves and how far it travels. On any given turn, the ball can move either one or two squares to the left, or one or two squares to the right. At the top of the screen are four boxes that show you which colors are linked to which directions. For instance, the leftmost box shows you which color makes the ball move one square to the left. The next box to the right shows you which color makes it move *two* squares to the left. The second pair of boxes show you which colors make the ball move in the opposite direction, to the right. By changing the color of the box where the ball is currently located, you can make it move toward your home square.

The playfield contains 11 boxes (9 in the TI version, 10 in the Atari ST version). When the game begins, each of these boxes is randomly given one of the four colors shown at the top of the screen. On each turn, you may change the color of one, several, or all of the boxes (however, you must always change at least one box). Below each box is a number which represents its distance from the home position of the player whose turn it is. For instance, if you are the player on the left, then on your turn the boxes are numbered 1, 2, 3, etc., from left to right (the tenth box is marked with a 0, and the eleventh with an A). When it's the right player's turn, the numbering is reversed (the rightmost box is 1, etc.).

To take a turn, you must select a number that corresponds to the numbers shown below the boxes in the playfield. This is done by pressing a single key. Press a number key from 1–0 to select one of the first ten values, or press the A key to choose the eleventh box. The number you choose determines how many boxes change color. For instance, if you press 1, only one box (the one nearest your home square) changes color. If you press 2, the two boxes nearest your home box change, and so on.

Where do the new colors come from? Every box cycles through the same series of four colors shown in the uppermost set of boxes, going from left to right. For example, if the colors shown there are white-blue-red-purple (the exact colors may be different on your computer), then a white square always changes to blue; a blue square always changes to red; a purple square changes to white, and so on. In other words, the box's current color determines which color it gets after the next color change.

Though every turn involves at least one color change, the ball doesn't necessarily move on every turn. It only moves when you change all the boxes between your home position and the current position of the ball. For example, if the ball is three boxes away from your home square, then you must change the color of at least three boxes in order to move it at all.

Foresight Rewarded

As you can see, there's much more to this game than appears on the surface. At first you might be tempted to try to move the ball as often as possible. But that's usually a losing strategy. Remember, the *direction* the ball moves depends on the color of its square before you take the turn.

In many cases, you'll want to move the ball only if it's on a color that moves it toward your goal. But like other games of strategy, Tug-a-War rewards the player who looks beyond the current move and tries to set things up for future moves; sometimes it's wise to make a small, temporary sacrifice in order to benefit later in the game. Because the boxes change colors in the same sequence, the effect of your own move is always completely predictable. However, since a single turn can change the color of many boxes, dramatic changes of fortune are also possible.

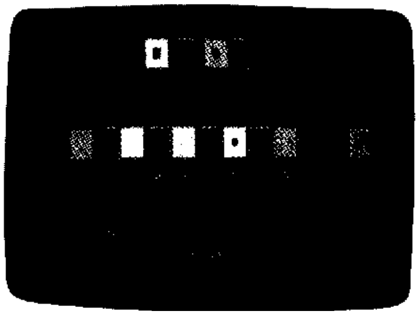
Amiga And 520ST Versions

Since the mouse is an integral system feature on both the Amiga and ST, both of these versions substitute mouse input for keyboard input. To select a square, simply move the mouse pointer to the desired box and press the left mouse button. Because keyboard prompts are unnecessary, no numbers are displayed below the playfield boxes.

Before entering BASIC to load the ST version, you should switch to the low-resolution graphics mode (use the Set Preferences option in the desktop's Options menu). Also, if your ST has 512K and a disk-based operating system, before running the program you should turn off buffered graphics (controlled by the Buffer Grph option in the Settings menu; it's off when no check appears beside the option in the menu). The standard 520ST leaves only about 5K free for BASIC programs, so Tug-a-War won't fit into memory unless the buffered graphics option is turned off. The program fits with buffered graphics switched on only if you have a 1040ST, or a 520ST with memory expansion, or a 520ST that has been upgraded with the TOS operating system in ROM chips (Read Only Memory).

The Amiga version uses the computer's built-in speech feature to announce the players' turns. In other respects, these games work exactly like the others.

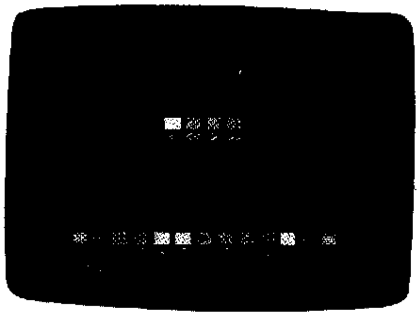
For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!.



"Tug-a-War" for the Commodore 64 and 128 is a game that looks simple, but demands good concentration and foresight.

Program 1: Tug-A-War For Commodore 64/128

```
JJ 100 POKE53280,0:PRINT"{CLR}
       ";:BC=53281:POKEBC,5:PC
       (1)=5:PC(2)=7:PS=6:CR=1
       :PL=1:X=20
CE 110 B=55715:A$="O{Y}P{DOWN}
       {3 LEFT}{H} {N}{DOWN}
       {3 LEFT}L{P}@":P$(1)="G
       REEN{RVS}{BLK} {OFF}
       {LEFT}":P$(2)="YELLOW":
       B$="{2 UP}"
DM 120 TM$="{HOME}{9 DOWN}":Q$
       ="{RVS}{BLK}{34 SPACES}
       ":DIMCL(11)
QG 130 FORI=0TO23:PRINT"{RVS}
       {BLK}{39 SPACES}"
KE 140 POKE1063+(40*I),160:POK
       E55335+(40*I),0:NEXT
AQ 150 PRINT"{RVS}{39 SPACES}
       {HOME}":POKE2023,160:PO
       KE56295,0
DG 160 PRINT"{HOME}"SPC(15)"
       {RVS}TUG-A-WAR":PRINTSP
       C(13)"{DOWN}{RVS}{WHT}"
       A$B$"{RED}"A$B$"{BLK}
       {CYN}"A$B$"{PUR}"A$B$
RS 170 PRINTSPC(14)"{BLK}1"SPC
       (2)"2"SPC(3)"1"SPC(2)"2
       {DOWN}":PRINTSPC(15)"
       {RVS}<C"SPC(5)"C>"
HK 180 GOSUB620:PRINTTM$"
       {DOWN}{RVS} {GRN}"A$B$;
       :FORZ=1TO11:Y=INT(4*RND
       (1))+1:CL(Z)=Y:POKE646,
       Y
FQ 190 PRINT"{RVS}"A$B$;:NEXTZ
       :PRINT"{YEL}{RVS}"A$
CB 210 POKE646,PEEK(B)AND15:PR
       INTTM$"{2 DOWN}"SPC(X)"
       {RVS}Q":POKEBC,PC(PL)
SF 220 PRINTTM$"{14 DOWN}"SPC(
       18)"{RVS}{BLK}
       {3 SPACES}"
HR 230 AN=0:PRINTTM$SPC(7)"
       {10 DOWN}{BLK}{RVS}HOW
       {SPACE}MANY TO CHANGE
       {OFF}"P$(PL)
MA 240 PRINTSPC(17)"{DOWN}
       {RVS}(1-A){2 DOWN}
       {3 LEFT}{I}{LEFT}";
BR 250 POKE204,0:POKE198,0:WAI
       T198,1:GETMT$
RB 260 IFASC(MT$)<48ORASC(MT$)
       <>65ANDASC(MT$)>57THEN2
       50
XR 270 POKE204,1:IFMT$="A"THEN
       AN=11:MT$="{LEFT}ALL":G
       OTO300
CR '280 IFMT$="0"THENAN=10:MT$=
       "10":GOTO300
HC 290 AN=VAL(MT$)
KM 300 PRINT"{2 LEFT}
       {5 SPACES}{3 LEFT}"MT$
?FS 310 IFAN<1ORAN>11THEN220
FF 320 IFPL=2THENAN=12-AN:GOTO
       440
HE 330 IFAN<PSTHENCK-1
KQ 340 FORQ=1TOAN:IFCL(Q)=4THE
       NCL(Q)=1:GOTO360
GE 350 CL(Q)=CL(Q)+1
KJ 360 NEXTQ:PRINTTM$:PRINT"
       {BLK}{RVS} {GRN}"A$B$;:
       FORZ=1TO11:POKE646,CL(Z
       ):PRINT"{RVS}"A$B$;
QD 370 NEXTZ:PRINT"{YEL}{RVS}"
       A$:POKE646,PEEK(B)AND15
       :PRINT"{HOME}{11 DOWN}"
       SPC(X)"{RVS} "
FH 380 IFCK=1THENCK=0:GOTO400
MH 390 ONPEEK(B)AND15GOSUB490,
       500,510,520
BF 400 IFPS<1THENPL=1:WC=5:B=5
       5698:X=2:GOTO530
DP 410 IFPS>11THENPL=2:WC=7:B=
       55734:X=38:GOTO530
KD 420 IFPL=1THENPL=2:GOSUB640
       :GOSUB630:GOTO210
HE 430 PL=1:GOSUB640:PRINT"
       {10 UP}":GOSUB620:GOTO2
       10
XH 440 FORQ=ANTO11
DH 450 IFAN<1ORAN>11THEN220
BD 460 IFAN>PSTHENCK=1
GJ 470 IFCL(Q)=4THENCL(Q)-1:GO
       TO360
HB 480 CL(Q)=CL(Q)+1:GOTO360
JX 490 B=B+6:X=X+6:PS=PS+2:RET
       URN
PA 500 B=B-3:X=X-3:PS=PS-1:RET
       URN
XM 510 B=B-6:X=X-6:PS=PS-2:RET
       URN
BE 520 B=B+3:X=X+3:PS=PS+1:RET
       URN
FF 530 POKE646,PEEK(B)AND15:PR
       INT"{HOME}{11 DOWN}"SPC
       (X)"{RVS}Q":PRINTTM$"
       {10 DOWN}";
PG 540 FORE=1TO5:PRINTQ$:NEXT:
       GOSUB640:GOSUB640
FM 550 PRINT"{HOME}{11 DOWN}"S
       PC(11)P$(PL)" IS THE WI
       NNER":Z=WC:FORI=1TO11:P
       OKEBC,Z
CP 560 IFZ=0THENZ=WC:GOSUB610:
       NEXT
SJ 570 Z=0:GOSUB610:NEXT
BP 580 POKEBC,15:PRINTTM$SPC(1
       1)"{10 DOWN}{RVS}LIKE T
       O PLAY AGAIN{2 DOWN}
       {11 LEFT}{RVS}Y/N"
RA 590 POKE198,0:WAIT198,1:GET
       MT$:IFMT$<>"N"THENRUN
JC 600 POKE198,0:SYS198
QC 610 FORP=1TO200:NEXTP:RETUR
       N
CG 620 PRINTTM$"{5 DOWN}{RVS}
       {BLK}{2 SPACES}<C 1
       {2 SPACES}2{2 SPACES}3
       {2 SPACES}4{2 SPACES}5
       {2 SPACES}6{2 SPACES}7
       {2 SPACES}8{2 SPACES}9
       {2 SPACES}0{2 SPACES}A"
       :RETURN
AA 630 PRINTTM$"{5 DOWN}{RVS}
       {BLK}{5 SPACES}A
       {2 SPACES}0{2 SPACES}9
       {2 SPACES}8{2 SPACES}7
       {2 SPACES}6{2 SPACES}5
```



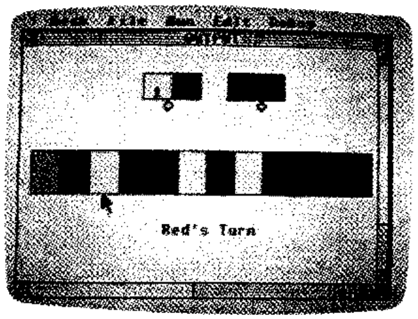
Apple "Tug-a-War," a challenging strategy game.

```
C9 170 HC = 6:PS = 12: GOSUB 43Ø
       : GOSUB 46Ø
F9 18Ø BP = 5: GOSUB 47Ø
69 19Ø VTAB 21: FOR I = 1 TO 11:
        HTAB I * 3 + 2: IF I < 1
       Ø THEN PRINT CHR$ (48 + I
       );
89 2ØØ IF I = 1Ø THEN PRINT "Ø";
15 21Ø IF I = 11 THEN PRINT "A";
E4 22Ø NEXT : VTAB 23: HTAB 1: P
       RINT "GREEN'S MOVE:";
46 23Ø GOSUB 52Ø:A = A - 1: IF (
       BP < = A) THEN BP = BP +
       JT(BC(BP))
97 24Ø FOR I = Ø TO A:BC(I) = BC
       (I) + 1 - 4 * (BC(I) = 3)
       :HC = CT(BC(I)):PS = I +
       1: GOSUB 43Ø: NEXT
54 25Ø GOSUB 47Ø
16 26Ø IF BP < Ø OR BP > 1Ø THEN
        36Ø
67 27Ø VTAB 21: FOR I = 1 TO 11:
        HTAB (12 - I) * 3 + 2: I
       F I < 1Ø THEN PRINT CHR$
       (48 + I);
99 28Ø IF I = 1Ø THEN PRINT "Ø";
25 29Ø IF I = 11 THEN PRINT "A";
80 3ØØ NEXT : VTAB 23: HTAB 1: P
       RINT "BLUE'S MOVE: ";
1F 31Ø GOSUB 52Ø:A = 11 - A: IF
       (BP > = A) THEN BP = BP +
        JT(BC(BP))
5A 32Ø FOR I = 1Ø TO A STEP - 1:
       BC(I) = BC(I) + 1 - 4 * (
       BC(I) = 3):HC = CT(BC(I))
       :PS = I + 1: GOSUB 43Ø: N
       EXT
51 33Ø GOSUB 47Ø
13 34Ø IF BP < Ø OR BP > 1Ø THEN
        36Ø
9F 35Ø GOTO 19Ø
92 36Ø PS = 12 * (BP > Ø) - 1: H
       COLOR= 4 * (BP > Ø): GOSU
       B 49Ø
33 37Ø VTAB 23: HTAB 1: IF BP <
       Ø THEN PRINT "GREEN WINS
         ": GOTO 39Ø
6C 38Ø IF BP > 1Ø THEN PRINT "BL
       UE WINS     "
0A 39Ø GET A$: GOTO 11Ø
DE 4ØØ FOR I = Ø TO 3: READ CT(I
       ): NEXT
3B 41Ø FOR I = Ø TO 3: READ JT(I
       ): NEXT : RETURN
84 42Ø DATA 3,5,6,2,-1,-2,1,2
8C 43Ø HCOLOR= HC: FOR YP = VP T
       O VP + 1Ø
9A 44Ø HPLOT PS * 21 + 1,YP TO P
       S * 21 + 17,YP: NEXT
1E 45Ø RETURN
AB 46Ø HCOLOR= 3: FOR YP = VP +
       1 TO VP + 9 STEP 2: HPLOT
        PS * 21 + 1,YP TO PS * 2
       1 + 17,YP: NEXT : RETURN
52 47Ø IF BP < Ø OR BP > 1Ø THEN
        RETURN
19 48Ø HCOLOR= 4 * (CT(BC(BP)) >
        3):PS = BP
74 49Ø FOR YP = VP + 3 TO VP + 7
       : HPLOT PS * 21 + 27,YP T
       O PS * 21 + 32,YP: NEXT :
        RETURN
DF 5ØØ TP = 124 + (TD + TN) * 21
        + TN * TX * 4:TL = TP +
       TD * 3:TR = TP - TD * 3
E5 51Ø HPLOT TR,6Ø TO TL,57 TO T
       R,54: RETURN
86 52Ø POKE 4916Ø,Ø: GET A$: IF
       A$ = CHR$ (3) THEN END
B9 53Ø IF A$ = CHR$ (3) THEN END
C8 54Ø IF A$ < > "A" AND A$ < >
       "a" AND (A$ < "Ø" OR A$ >
        "9") THEN 52Ø
36 55Ø IF A$ = "A" OR A$ = "a" T
       HEN A$ = CHR$ (59)
38 56Ø IF A$ = "Ø" THEN A$ = CHR
       $ (58)
51 57Ø A = ASC (A$) - 48: RETURN
```

Program 5: Tug-A-War For Atari ST

Version by Kevin Mykytyn, Editorial Programmer

```
10 fullw 2:clearw 2:color 1,1,1
20 bp=6:c(1)=6:c(2)=7:c(3)=10:c(4)=12:pl
   =0
30 mov(1)=-1:mov(2)=-2:mov(3)=1:mov(
   4)=2
40 gosub 170:gosub 270
50 pl=(pl=0):gosub drawball:gosub play
   er
60 gosub readmouse:if y<98 or y>127 o
   r x<34 or x>273 then 60
70 sp=int((x-11)/24)
80 if (pl=0 and sp>bp) or (pl=-1 and sp
   <bp) then 110
90 t=c(col(bp)):color 1,t,t:pcircle bp*24+2
   2,91,6
100 bp=bp+mov(col(bp)):if bp>11 the
    n bp=11 else if bp=-1 then bp=0
110 gosub colchange
120 gosub drawball:if bp>0 and bp<11 th
    en 50
130 gotoxy 13,14:if bp=0 then color 5:pri
    nt " Blue Wins! ":goto 150
140 color 2:print " Red Wins! "
150 gotoxy 10,16:color 1:print "Press Mou
    se Button":gosub readmouse:clear: go
    to 10
160 drawball: color 1,1,1:pcircle bp*24+2
    2,91,6:return
170 for a=75 to 105 step 30:linef 10,a,298,
    a:next
180 for a=10 to 298 step 24:linef a,75,a,10
    5:next
190 color 1,5:fill 12,77:color 1,2:fill 296,77
200 gotoxy 12,3:print "1 2 1 2"
210 for a=100 to 220 step 24:linef a,19,a,3
    7:next
220 for a=19 to 37 step 18:linef 100,a,148,
    a:linef 172,a,220,a:next
230 color 1,6:fill 101,20:color 1,7:fill 125,20
240 color 1,10:fill 173,20:color 1,12:fill 197,
    20
250 gotoxy 13,4:print chr$(4);" ";chr$(3)
260 return
270 for a=1 to 10:q=int(rnd(1)*4+1):col(a
    )=q:color 1,c(q):fill 25+a*24,77
280 next:return
290 readmouse: poke contrl,124
300 poke contrl+2,0:poke contrl+6,0
310 vdisys(0):if peek(intout)=0 then 310
320 x=peek(ptsout):y=peek(ptsout+2)
330 return
340 colchange: if pl=0 then 360
350 for a=1 to sp:gosub 370:next:return
360 for a=10 to sp step-1:gosub 370:next:r
    eturn
370 col(a)=col(a)+1+4*(col(a)=4)
380 color 1,c(col(a)):fill 25+a*24,77
390 return
400 player: gotoxy 13,14:if pl=0 then colo
    r 2:print "Red's Turn ":return
410 color 5:print "Blue's Turn":return
```



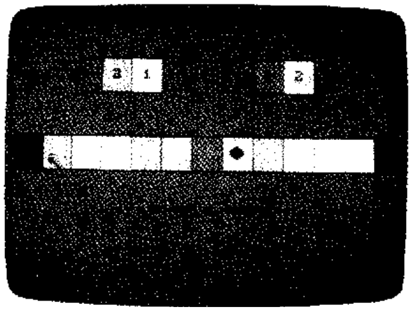
Use the mouse to play the Atari ST version of "Tug-a-War."

Program 6: Tug-A-War For TI-99/4A

Version by Patrick Parrish, Programming Supervisor

```
100 GOTO 150
110 FOR I=1 TO LEN(A$)
120 CALL HCHAR(R,C+I,ASC(SE
    G$(A$,I,1)))
130 NEXT I
140 RETURN
150 RANDOMIZE
160 CALL COLOR(14,1,7)
170 CALL SCREEN(2)
180 PC(0)=5
190 PC(1)=7
200 P$(0)="BLUE"
210 P$(1)="RED"
220 Y$(0)="<- 1  2  3  4  5
      6  7  8  9     "
230 Y$(1)="     9  8  7  6  5
      4  3  2  1 ->"
240 KHAR(0)=0
250 KHAR(10)=5
260 FOR I=96 TO 136 STEP 8
270 CALL CHAR(I,"0000000000
    000000")
280 CALL CHAR(I+1,"0F0F0F0F
    0F0F0F0F")
290 CALL CHAR(I+2,"3078FCFC
    FC7830")
300 CALL CHAR(I+3,"00103010
    10101038")
310 CALL CHAR(I+4,"00384404
    0810207C")
320 NEXT I
330 PS=5
340 PL=0
350 BP=17
360 CALL CLEAR
370 GOSUB 1000
380 PRINT TAB(11);"TUG-A-WA
    R"
390 PRINT : :
400 B$=CHR$(128)&CHR$(128)&
    CHR$(129)
410 PRINT TAB(9);"hhippq xx
    y";B$
420 PRINT TAB(9);"hkiptq x{
    y";CHR$(128);CHR$(132);
    CHR$(129)
430 PRINT TAB(9);"hhippq xx
    y";B$
```

```
440 PRINT
450 PRINT TAB(11);"<-";TAB(
    18);"->"
460 FOR I=1 TO 15
470 PRINT
480 NEXT I
490 FOR I=1 TO 9
500 RANDOMIZE
510 KHAR(I)=INT(4*RND)+1
520 NEXT I
530 FOR R=13 TO 15
540 CALL HCHAR(R,2,96,2)
550 FOR I=1 TO 9
560 KH=96+KHAR(I)*8
570 CALL HCHAR(R,I*3+1,KH)
580 CALL HCHAR(R,I*3+2,KH)
590 CALL HCHAR(R,I*3+3,KH+1
    )
600 NEXT I
610 CALL HCHAR(R.31.136.2)
620 NEXT R
630 CALL HCHAR(14,BP,96+KHA
    R(PS)*8+2)
640 IF (PS=0)+(PS=10)THEN 1
    180
650 A$=Y$(PL)
660 R=17
670 C=1
680 GOSUB 110
690 CALL HCHAR(24,17,32)
700 A$=P$(PL)&"'S TURN "
710 R=20
720 C=11
730 GOSUB 110
740 R=22
750 C=14
760 A$="(1-9)"
770 GOSUB 110
780 GOSUB 1020
790 CALL KEY(0,K,H)
800 IF H=0 THEN 790
810 IF (K<49)+(K>57)THEN 79
    0
820 AN-K-48
830 CALL HCHAR(24,17,K)
840 IF PL=0 THEN 890
850 AN=10-AN
860 S=AN
870 E=9
880 GOTO 910
890 S=1
900 E=AN
910 GOSUB 1100
920 FOR Q=S TO E
930 IF KHAR(Q)<>4 THEN 960
940 KHAR(Q)=1
950 GOTO 970
960 KHAR(Q)=KHAR(Q)+1
970 NEXT Q
980 PL=-(PL=0)
990 GOTO 530
1000 CALL COLOR(9,1,5)
1010 CALL COLOR(14,1,7)
1020 FOR I=1 TO 8
1030 CALL COLOR(I,PC(PL),2)
1040 NEXT I
1050 CALL COLOR(10,PC(PL),1
     6)
1060 CALL COLOR(11,PC(PL),1
     1)
1070 CALL COLOR(12,PC(PL),8
     )
1080 CALL COLOR(13,PC(PL),1
     4)
1090 RETURN
1100 IF ((AN<PS)*(PL=0))+(A
     N>PS)*(PL=1)THEN 1170
1110 A=(KHAR(PS)=1)+(KHAR(P
     S)=2)*2-(KHAR(PS)=3)-(
     KHAR(PS)=4)*2
1120 BP=BP+A*3
1130 PS=PS+A
1140 IF (PS>0)*(PS<10)THEN
     1170
1150 PS=-(PS=-1)+(PS=11)+PS
1160 BP=-(BP<5)*3-(BP>29)*3
     1
1170 RETURN
1180 R=14
1190 C=7
1200 A$=P$(-(PS=10))&" IS T
     HE WINNER!"
1210 GOSUB 110
1220 A$="LIKE TO PLAY AGAIN
      (Y/N)?"
1230 R=24
1240 C=4
1250 GOSUB 110
1260 CALL KEY(0,K,H)
1270 IF H=0 THEN 1260
1280 IF (K<>78)*(K<>89)THEN
      1260
1290 IF K=89 THEN 330
```



This version of "Tug-a-War" uses several of the Amiga's 4,096 different color shades.

Program 7: Tug-A-War For Amiga

Version by John Krause, Assistant Technical Editor

```
SAY TRANSLATE$("")←
SCREEN 2,320,200,3,1←
WINDOW 2," Tug-A-War ",,12,2←
FOR i=0 TO 7←
 READ r,g,b←
 PALETTE i,r,g,b←
NEXT←
RANDOMIZE TIMER←
DIM a(11)←
FOR i=1 TO 11←
 a(i)=INT(RND(1)*4)+4←
NEXT←
row=3←
col=3:colr=4:GOSUB frame:GOSUB squ
are←
col=4:colr=5:GOSUB frame:GOSUB squ
are←
col=8:colr=6:GOSUB frame:GOSUB squ
are←
col=9:colr=7:GOSUB frame:GOSUB squ
are←
row=10←
LOCATE 5,11:COLOR 1,4:PRINT "2"←
LOCATE 5,14:COLOR 1,5:PRINT "1"←
LOCATE 5,26:COLOR 1,6:PRINT "1"←
LOCATE 5,29:COLOR 1,7:PRINT "2"←
LINE (64,36)-STEP(-16,0),1←
LINE -STEP(8,4),1←
LINE (48,36)-STEP(8,-4),1←
LINE (248,36)-STEP(16,0),1←
LINE -STEP(-8,4),1←
LINE (264,36)-STEP(-8,-4),1←
FOR col=0 TO 12←
 GOSUB frame←
NEXT←
col=0:colr=3:GOSUB square←
col=12:colr=2:GOSUB square←
dot=6:GOSUB update←
SAYTRANSLATE$("welcometotugowa
r.")←
main:←
LOCATE 17,15←
IF red THEN←
 COLOR 2,0:PRINT "Red's turn "←
 SAY TRANSLATE$("reds turn.")←
ELSE←
 COLOR 3,0:PRINT "Blue's turn"←
 SAY TRANSLATE$("blues turn.")←
END IF←
WHILE MOUSE(0)<>1 OR MOUSE(4)<8
0 OR MOUSE(4)>104 OR MOUSE(3)<2
3 OR MOUSE(3)>276←
WEND←
click=INT(MOUSE(3)/24)←
IF (red AND click<=dot) OR (red=0 AN
D click>=dot) THEN ←
 temp=dot←
 IF a(temp)=4 THEN dot=dot-2←
 IF a(temp)=5 THEN dot=dot-1←
 IF a(temp)=6 THEN dot=dot+1←
 IF a(temp)=7 THEN dot=dot+2←
END IF←
IF red THEN←
 FOR i=click TO 11←
  a(i)=a(i)+1←
  IF a(i)=8 THEN a(i)=4←
  NEXT←
ELSE←
 FOR i=1 TO click←
  a(i)=a(i)+1←
  IF a(i)=8 THEN a(i)=4←
  NEXT←
END IF←
IF dot>11 THEN←
 dot=12:GOSUB update←
 LOCATE 17,15:COLOR 2,0:PRINT " Re
d wins! "←
 SAY TRANSLATE$("red wins.")←
 GOTO quit←
END IF←
IF dot<1 THEN←
 dot=0:GOSUB update←
 LOCATE 17,15:COLOR 3,0:PRINT "Blu
e wins! "←
 SAY TRANSLATE$("blue wins.")←
 GOTO quit←
END IF←
GOSUB update←
red=1-red←
GOTO main←
frame:←
x=24:IF 24*col>280 THEN x=23←
LINE (24*col,8*row)-STEP(x,24),1,b←
RETURN←
square:←
x=22:IF 24*col+1>280 THEN x=21←
LINE (24*col+1,8*row+1)-STEP(x,22),c
olr,bf←
RETURN←
update:←
FOR col=1 TO 11←
 colr=a(col):GOSUB square←
NEXT←
CIRCLE (24*dot+11,91),5,1←
PAINT (24*dot+11,91),1←
RETURN ←
quit:←
LOCATE 19,7:COLOR 1,0:PRINT "Clic
k mouse to play again."←
SAY TRANSLATE$("click mouse to pla
y again.")←
WHILE MOUSE(0)=0:WEND←
RUN←
DATA .5,.5,.5,0,0,0,1,0,0,0,0,1,0,1,0,1,1,
0,1,0,1,0,1,1←
```

The Beginners Page

Tom R. Halfhill, Editor

More String-Slicing

Last month we saw how you can copy pieces of character strings using the LEFT$ and RIGHT$ functions found in versions of Microsoft BASIC. For even more flexibility, most Microsoft-style BASICs include a third function for extracting sections of strings. Called MID$ ("mid-string"), this function lets you copy a section from the middle of a string.

The basic format is MID$ (*string$,n1,n2*), where *string$* is a string variable or literal string; *n1* is a number representing the beginning character position of the substring you want to extract; and *n2* is a number representing the number of characters in the substring you want to extract. For example:

```
10 A$="JAMES FENIMORE COOPER"
20 PRINT MID$(A$,7,8)
30 B$=MID$(A$,11,4)
40 PRINT B$
50 PRINT A$
```

When you run this program, the result is:

```
FENIMORE
MORE
JAMES FENIMORE COOPER
```

Line 20 prints the eight characters starting at position seven in A$, resulting in the substring FENIMORE. (Remember that spaces count as characters.) Lines 30 and 40 do much the same thing, but copy the four characters starting at position 11 into the string variable B$ before printing them out. This method is useful if you need to print B$ later in your program or manipulate B$ in some other way. Line 50 shows that the MID$ function, like LEFT$ and RIGHT$, does not disturb the original contents of A$.

MID$ is handy for so many different things that it's hard to come up with a generalized example. It can even be used to replace LEFT$ and RIGHT$—for instance, MID$(A$,1,10) is equivalent to LEFT$(A$,10), and MID$(A$,LEN(A$)−9,LEN(A$)) is the same as RIGHT$(A$,10). One useful application of MID$ is to store a bunch of short strings as a single long string, then pick out the substring you want with MID$. For example, let's say you're writing some sort of program that needs to print out the months of the year, perhaps as labels for a budget or chart. You could abbreviate the names of the months as equal-length substrings within one large string, like this:

```
10 M$="JANFEBMARAPRMAYJUNJUL
  AUGSEPOCTNOVDEC"
```

Now suppose that the numeric variable M contains the number of the month you need to print out—maybe as a result of an INPUT statement:

```
20 PRINT "WHICH MONTH TO
  PRINT";
30 INPUT M
40 PRINT MID$(M$,M*3-2,3)
```

Depending on the user's response (1 for January, 2 for February, etc.), line 40 prints out the proper month name. Or you could print out all the months with a loop—FOR M=1 TO 12:PRINT MID$(M$,M*3−2,3) :NEXT M.

Storing all the months in a single string and extracting the one you want with MID$ is more efficient than using 12 separate strings for the same purpose. It's also more efficient in some ways than a string array (a subject we'll cover in a future column).

Atari And TI Strings

There are no LEFT$, RIGHT$, or MID$ functions in TI BASIC or the Atari BASIC found on Atari 400/800, XL, and XE computers. These BASICs handle strings a little differently than Microsoft BASIC does. (Note that Microsoft BASIC is available on cartridge for Atari computers, and some BASICs available from independent suppliers also support Microsoft-style strings.)

TI BASIC's statement for segmenting strings is SEG$. It works exactly like MID$ in Microsoft BASIC—the statement B$=SEG$ (A$,11,4) is equivalent to B$=MID$(A$,11,4). You can simulate LEFT$ with a statement in the form SEG$(*string$,n1,n2*), where *string$* is the string you wish to manipulate, *n1* is the starting character position of the segment within the string, and *n2* is the number of characters you wish to print or copy. For example, the statement B$=LEFT$(A$,6) can be replaced with B$=SEG$(A$,1,6).

Simulating RIGHT$ is a bit more complicated. You need a statement in the form SEG$(*string$*,LEN(*string$*)−*n1*,*n2*), where *n2* is the number of characters you wish to print or copy, and *n1* is *n2*−1. For example, B$=RIGHT$(A$,6) can be replaced with B$=SEG$ (A$,LEN(A$)−5,6).

Atari BASIC requires the same sort of manipulations. To print or copy any substring in Atari BASIC, simply specify the starting and ending character positions of the substring within the larger string. To translate B$=LEFT$(A$,6), use B$=A$(1,6). To simulate RIGHT$, use a statement in the form *string$*(LEN(*string$*)−*n*,LEN(*string$*)), where *string$* is the string you're manipulating and *n* is the number of characters you wish to print or copy minus one. For instance, to translate B$=RIGHT$(A$,6), use B$=A$(LEN(A$)−5,LEN(A$)). To simulate MID$, use the statement *string$*(*n1*,*n2*), where *n1* is the starting character position (just like MID$), and *n2* equals *n1* plus the number of characters you wish to print or copy minus one. Thus, the Microsoft statement B$=MID$ (A$,11,4) is translated as B$=A$ (11,14). ©

Programming the TI

C. Regena

Creating Rhythms

A year ago, in the March 1985 issue of COMPUTE!, I published a program called "Drum Practice" for the TI-99/4A. That program was limited to quarter notes and quarter rests and the rhythms listed in DATA statements. This month, I'm offering a more complex program.You can create the rhythm for one measure by choosing notes and rests, and then the computer will play the rhythm for eight measures.

Lines 110–190 print the instructions. The different kinds of notes and rests available will appear at the bottom of the screen. You can use the arrow keys (on S and D) to move the red marker left or right to make your selection, then press the ENTER key. Your choice will then be printed on the staff above.

The available notes are a quarter note, two eighth notes together, one eighth note, two sixteenth notes, a dotted eighth note with a sixteenth note, a quarter rest, and an eighth rest.

Line 200 sets the time T equal to 75. If you want the rhythm to play faster decrease this number which represents the duration of a sixteenth note. Lines 210–240 read in from data (lines 260–330) the definitions for the graphic characters and define the characters from numbers 91 to 128. Line 340 defines R for a row number for the staff. Lines 350–360 define the red arrow used as a marker under the notes to be selected.

Lines 370–390 define variables in an array for the seven possible choices. Line 400 contains the data for this loop. For each of the choices from 1 to 7, D(C) is a value representing the counts—4 for one count, 2 for a half count. This variable is used to make sure the user makes a valid choice. For example, the computer will not allow a quarter note to be chosen if only a half of a count is left in the measure. COL(C) is the column and is used to place the red marker.

The Rhythm Track

S$(C) represents the durations when the rhythm is played. A sixteenth note factor is 1, so the quarter note is 4. Two eighth notes are 22, and one eighth note is 2. The two sixteenth notes are 11, and a dotted eighth with a sixteenth are 31. The rests are W and H. As the notes and rests are chosen, the string RHY$ will add on values of S$ (line 870).

Lines 1250–1390 play the rhythm. Line 1270 finds the length L of the string RHY$. Line 1280 starts the loop for L number of times. Line 1290 looks at one character at a time of RHY$. If the character A$ is a letter, a rest is indicated so a frequency of 9999 with a volume of 30 is used. If A$ is a number, that number is used as a factor times the previously defined T for the duration in the CALL SOUND statement, line 1310. If you prefer a different sound, change the frequency numbers in line 1310. I used the noise of −5 plus the frequency of 330. Line 1370 stops the sound so you can hear the different notes. The measure is played eight times.

Lines 420–440 wait for you to press ENTER before the program continues. Lines 460–500 clear the screen and print the notes using the redefined symbols. The lowercase y and z are typed by releasing the ALPHA LOCK key. Most of the symbols are typed by using the function key. Lines 510–630 draw the staff.

Lines 650–690 initialize variables for choosing the notes. COUNT and CHECK are used to determine how many notes and rests can be used in the measure. This measure is 4/4 time. A sixteenth note has a value of 1, so the COUNT will go up to 16. CHECK is how many points are remaining in the measure. These numbers are used to verify which notes and rests can be used in the measure.

PLACE is the column number where the note or rest will start being drawn on the staff. The first note will start in column 8. PLACE is incremented depending on which note or set of notes is chosen. Lines 690–820 are the lines to get the user's choice. Line 730 makes sure the left arrow key (S), the right arrow key (D), or the ENTER key is chosen; all other keys are ignored by branching back to the CALL KEY statement. C is the choice number, and COL(C) is the column where the red marker appears for the choice.

It's Timing That Counts

Line 830 makes sure the choice is valid. The D timing value must be less than or equal to the number of sixteenth counts available. If the choice is not valid, the program plays an "uh-oh" sound and branches back to line 720, which is the CALL KEY statement to get another choice. Line 870 increments the RHY$ string with the appropriate timing factors. Line 880 branches to the proper place for drawing the notes or rest and incrementing PLACE.

Lines 1220–1240 increment the COUNT and recalculate the CHECK time. If the measure is not full, the program goes back to get another choice. Lines 1250–1390 play the measure eight times. To stop the program, press FCTN BREAK.

If you have trouble running this program and get an error message in 220, 230, or 380, the actual cause of the error is most likely in the DATA statements of lines 260–330 or line 400.

All notes are placed at the E space of the staff, representing a snare drum rhythm. You may add to the program by including bass drum notes, cymbal rhythms, and tom-toms. To use this program for a melody instrument, you can use the up and down keys to move the note on the staff, then use a variable frequency to play the note.

You may use the general idea of this program in choosing items to go with a different theme of graphics, not music—perhaps building a game or drawing a picture by choosing different shapes.

Rhythms

```
100 REM  RHYTHMS
110 CALL CLEAR
120 PRINT TAB(1Ø);"RHYTHMS"
130 PRINT ::"CHOOSE THE NOT
    ES FROM THE"
140 PRINT :"BOTTOM OF THE S
    CREEN BY"
150 PRINT :"USING THE LEFT
    AND RIGHT"
160 PRINT :"ARROW KEYS TO M
    OVE AND"
170 PRINT :"THE ENTER KEY T
    O SELECT."
180 PRINT ::"WHEN THE MEASU
    RE IS COMPLETE"
190 PRINT :"YOU WILL HEAR T
    HE RHYTHM."
200 T=75
210 FOR C=91 TO 128
220 READ C$
230 CALL CHAR(C,C$)
240 NEXT C
250 REM  DATA FOR CHARACTER
    S
260 DATA ØØØØØFØ8ØFØ8Ø8Ø8,Ø
    ØØØF8Ø8F8Ø8Ø8Ø8,Ø8Ø8Ø83
    8F9E8,ØØØØFFØ1Ø7Ø1Ø1Ø1,
    1ØØ8ØC183Ø3ØØ8Ø4,1C2Ø2Ø
    1ØØ8
270 DATA ØØØØØØØØØØØØFFDØ,D
    ØDØFFDØD4DØFFDØ,D4DØFFD
    ØDØDØFF,ØØØØØØØØØØØØØFF,
    ØØØØFFØØØØØØFF,ØØØØØØØØ
    ØØØØFFØB
280 DATA ØBØBFFØB2BØBFFØB,2
    BØBFFØBØBØBFF,ØØØØØØØØ
    ØØØFF1Ø,2424447EØ4Ø4FF1
    ,1Ø2424447EØ4FF,ØØØØØ8Ø
    8Ø8Ø8FFØ8
290 REM
300 DATA Ø8Ø8FF78F87ØFF,ØØØ
    ØØFØ8Ø8Ø8FFØ8,ØØØØF8Ø8Ø
    8Ø8FFØ8,ØØØØØ8ØCØAØ9FFØ
    8,ØØØØØFØ8ØFØ8FFØ8
310 DATA ØØØØF8Ø8F8Ø8FFØ8,Ø
    ØØØFFØ1Ø7Ø1FFØ1,ØØØØFF6
    27EØ4FF1,1Ø2ØFFØØØØØØFF
    ,Ø8Ø8FF78F978FF,1ØØ8ØC1
    83Ø3ØØ8Ø4
320 DATA 1C2Ø2Ø1ØØ8,ØØØØØ8Ø
    8Ø8Ø8Ø8Ø8,Ø8Ø8Ø878F87,Ø
    ØØØØFØ8Ø8Ø8Ø8Ø8,ØØØØF8Ø
    8Ø8Ø8Ø8Ø8,ØØØØØ8ØCØAØ9Ø
    8Ø8
330 DATA 627EØ2Ø4Ø81Ø2,Ø1Ø1
    FFØF1FØEFF,Ø1Ø1Ø1ØF1FØE
340 R=5
350 CALL CHAR(136,"1Ø387CFE
    1Ø1Ø1Ø1")
360 CALL COLOR(14,7,1)
370 FOR C=1 TO 7
380 READ D(C),COL(C),S$(C)
390 NEXT C
400 DATA 4,4,4,4,8,22,2,12,
    2,2,15,11,4,19,31,4,24,
    W,2,28,H
410 REM
420 PRINT ::"PRESS <ENTER>
    TO START."
430 CALL KEY(Ø,K,S)
440 IF K<>13 THEN 43Ø
450 REM  DRAW NOTES
460 CALL CLEAR
470 PRINT " y   {!  }  [\
    {^   _"
480 PRINT " z   zz  z  zz
    zz   `   ~"
490 CALL HCHAR(23.2Ø,128)
500 PRINT
510 REM  DRAW STAFF
520 CALL HCHAR(R,3,97)
530 CALL HCHAR(R  ,3,98)
540 CALL HCHAR(R  ,3,99)
550 CALL HCHAR(R,4,1ØØ,22)
560 CALL HCHAR(R+1,4,1Ø1,22
    )
570 CALL HCHAR(R+2,4,1Ø1,22
    )
580 CALL HCHAR(R,26,1Ø2)
590 CALL HCHAR(R+1,26,1Ø3)
600 CALL HCHAR(R+2,26,1Ø4)
610 CALL HCHAR(R,5,1Ø5)
620 CALL HCHAR(R+1,5,1Ø6)
630 CALL HCHAR(R+2,5,1Ø7)
640 REM  CHOOSE NOTES
650 COUNT=Ø
660 CHECK=16
670 PLACE=8
680 RHY$=""
690 C=1
700 CALL HCHAR(23,COL(C),13
    6)
710 CALL SOUND(1ØØ,14ØØ,2)
720 CALL KEY(Ø,K,S)
730 IF (K<>13)*(K<>83)*(K<>
    68)THEN 72Ø
740 CALL HCHAR(23,COL(C),32
    )
750 IF K<>83 THEN 78Ø
760 C=C-1
770 IF C>=1 THEN 7ØØ ELSE 6
    9Ø
780 IF K<>68 THEN 83Ø
790 C=C+1
800 IF C<=7 THEN 7ØØ
810 C=7
820 GOTO 7ØØ
830 IF D(C)<=CHECK THEN 87Ø
840 CALL SOUND(15Ø,33Ø,2)
850 CALL SOUND(15Ø,262,2)
860 GOTO 7ØØ
870 RHY$=RHY$&S$(C)
880 ON C GOTO 89Ø,93Ø,99Ø,1
    Ø3Ø,1Ø9Ø,115Ø,119Ø
890 CALL HCHAR(R,PLACE,1Ø8)
900 CALL HCHAR(R+1,PLACE,1Ø
    9)
910 PLACE=PLACE+3
920 GOTO 122Ø
930 CALL HCHAR(R,PLACE,11Ø)
940 CALL HCHAR(R+1,PLACE,1Ø
    9)
950 CALL HCHAR(R,PLACE+1,11
    1)
960 CALL HCHAR(R+1,PLACE+1,
    1Ø9)
970 PLACE=PLACE+3
980 GOTO 122Ø
990 CALL HCHAR(R,PLACE,112)
1000 CALL HCHAR(R+1,PLACE,1
     Ø9)
1010 PLACE=PLACE+2
1020 GOTO 122Ø
1030 CALL HCHAR(R,PLACE,113
     )
1040 CALL HCHAR(R+1,PLACE,1
     Ø9)
1050 CALL HCHAR(R,PLACE+1,1
     14)
1060 CALL HCHAR(R+1,PLACE+1
     ,1Ø9)
1070 PLACE=PLACE+2
1080 GOTO 122Ø
1090 CALL HCHAR(R,PLACE,11Ø
     )
1100 CALL HCHAR(R+1,PLACE,1
     18)
1110 CALL HCHAR(R,PLACE+1,1
     15)
1120 CALL HCHAR(R+1,PLACE+1
     ,127)
1130 PLACE=PLACE+3
1140 GOTO 122Ø
1150 CALL HCHAR(R+1,PLACE,1
     19)
1160 CALL HCHAR(R+2,PLACE,1
     2Ø)
1170 PLACE=PLACE+3
1180 GOTO 122Ø
1190 CALL HCHAR(R+1,PLACE,1
     16)
1200 CALL HCHAR(R+2,PLACE,1
     17)
1210 PLACE=PLACE+2
1220 COUNT=COUNT+D(C)
1230 CHECK=16-COUNT
1240 IF COUNT<16 THEN 7ØØ
1250 REM  PLAY RHYTHM
1260 FOR TIME=1 TO 8
1270 L=LEN(RHY$)
1280 FOR M=1 TO L
1290 A$=SEG$(RHY$,M,1)
1300 IF (A$="W")+(A$="H")TH
     EN 133Ø
1310 CALL SOUND(T*VAL(A$),-
     5,2,33Ø,4)
1320 GOTO 137Ø
1330 REST=T
1340 IF A$="H" THEN 136Ø
1350 REST=REST*2
1360 CALL SOUND(REST,9999,3
     Ø)
1370 CALL SOUND(1,9999.3Ø)
1380 NEXT M
1390 NEXT TIME
1400 FOR DEL=1 TO 5ØØ
1410 NEXT DEL
1420 GOTO 46Ø
1430 END                    ©
```

News & Products

Interactive Science Fiction Game For Commodore, Apple

PSI-5 Trading Company, from Accolade Software, is a science fiction "mini-drama," whose plot and outcome are contingent on the player's relationship with the story's characters. The game features detailed graphic depictions of 30 different characters who interact with the player through conversational text.

The story revolves around the PSI-5 Trading Company, a space freighter setting off on a mission to save the inhabitants of the Parvin Frontier from alien invaders. As captain of the ship, you must choose a crew of 5 from 30 applicants, each possessing special skills and a unique personality. The success of the mission hinges on the confidence you have in your crew to handle its responsibilities.

Suggested retail price for the Commodore 64 version is $29.95. The Apple version retails for $34.95.

Accolade Software, 20863 Stevens Creek Blvd., Cupertino, CA 95014.

Circle Reader Service Number 212.

Epyx Games Available For Amiga And Atari ST

Epyx has announced that two of its most popular computer games, and a microcomputer version of a classic mainframe adventure game, will be available for the Amiga and Atari ST this spring. In *Winter Games,* up to eight people can compete in seven events from the Winter Olympics. The original Commodore version of the game featured excellent graphics and sound. *Rogue* was originally a mainframe computer adventure game often played on college campuses. And the *Temple of Apshai Trilogy* offers a wide range of multiple dungeon levels, featuring 1400 separate chambers, plus enhanced high-resolution graphics.

Both versions of all three games are expected to retail for between $19–$39.

Epyx, Inc., 1043 Kiel Ct., Sunnyvale, CA 94089.

Circle Reader Service Number 213.

Bantam Software Promotions

Bantam Electronic Publishing is offering software promotional deals for purchasers of *Sherlock Holmes In "Another Bow," The Fourth Protocol,* and *The Complete Scarsdale Medical Diet.* Through April 15, special rebate coupons can be used to take $5 off the price of each of those programs. And, through March 31, Bantam will take entries in its Mystery Weekend contest, the winner of which will get a weekend for two in Boston to participate in a "mystery weekend" at the famous Parker House hotel. Special Holmes mystery pamphlets are available in many participating software stores. The pamphlets contain a mystery which you solve, and then submit to Bantam for a drawing in mid-April.

Bantam Electronic Publishing, Bantam Books, 666 Fifth Ave., New York, NY 10103.

Circle Reader Service Number 214.

Sports Tutorials Designed By The Pros

Avant-Garde has enlisted the help of three famous professional athletes in developing a line of sports tutorials. *Joe Theismann's Pro Football* offers advice on training and strategy to help develop quarterbacking techniques; improves overall football skills; and helps you understand the finer points of the game. Plays are illustrated through live-action diagrams. *Dave Winfield's Batter Up!* advises on pitchers, batting stance, swing height, grip and hitting strategy to help you develop expert batting techniques. The package also includes Winfield's book, *Batter Up! The Act of Hitting,* and a four-player batting game, *Slugfest!. Chris Evert-Lloyd Tennis* provides animated demonstrations of grip, stroke, game strategy, and specialized exercises. The program helps you learn the rules and choose the best equipment, and teaches concentration techniques to prepare you mentally for a match.

The Commodore 64 version of each program retails for $34.95. The Apple II version (64K RAM minimum) and IBM-PC/PCjr version (128K RAM minimum) retail for $39.95.

Avant-Garde, 37B Commercial Blvd., Novato, CA 94947.

Circle Reader Service Number 215.

Infocom At The Big Top

In *Ballyhoo,* Infocom's new interactive mystery, you are a small-town circusgoer who sticks around after the show to explore the exotic back lot. What you discover is a mysterious underworld of crime and corruption, into which the circus owner's daughter has been kidnapped. In order to find her, you must solve a series of puzzles that are hidden among the circus folk.

Ballyhoo, one of Infocom's standard-level, all-text adventure games, is available for the Apple II-series and Macintosh; Atari XL/XE and ST series, Commodore 64/128 and Amiga, and the IBM PC and PCjr, for a list price of $39.95.

Infocom, Inc., 125 Cambridge Park Dr., Cambridge, MA 02140.

Circle Reader Service Number 216.

New From Better Working

Word Processor with Spellchecker is the third product to be released in Spinnaker's Better Working line of home productivity software. The program is a full-function word processor, with a 50,000 word American Heritage Dictionary to catch spelling mistakes. It also features a 750-word personalized user dictionary, preview mode, microcommands for alternative print styles, and window-based menus and help screens.

The other titles in the Better Working series are *Spreadsheet* and *File and Report. Word Processor with Spellchecker* can perform mailmerge with *Better Working File and Report.* Each program is available for the Apple II series ($59.95) and the Commodore 64/128 ($49.95).

Better Working, Spinnaker, One Kendall Square, Cambridge, MA 02139.

Circle Reader Service Number 217.

©

Classified

SOFTWARE

TI-99/4A Software/Hardware bargains. Hard-to-find items. Huge selection. Fast service. Free catalog. D.E.C., Box 690, Hicksville, NY 11801

LOTTO PICKER. Improve your chances for those Million Dollar Jackpots! Picks LOTTO, WIN-4, & Daily Numbers. All USA & Can. games incl. Expandable! IBM/C64/TI99 $29.95. Order Now! 1-800-341-1950 Ext. 77. Mail Orders: Ridge, 170 Broadway, #201-C, NYC, NY 10038. Catalog.

PROJECT PLANNING/MANAGEMENT using the C64, SX, or C128. Data sheet for SASE - Prgm for $106.95 (CA res. add 6% sls tx). LAWCO, Dept. C, Box 2009, Manteca, CA 95336

Genealogy Program for the C64. **"FAMILY TREE"** will produce Pedigree Charts, Family Group Records, Individual Files, Indexes, Searches of Ancestors. LDS version available. "The Best" genealogy program for the 64. $49.95, GENEALOGY SOFTWARE, POB 1151, PORT HURON, MI 48061, (519) 344-3990.

Animal Records maintained with **"PETIGREE"** for the C64. Produces Litter, Awards, Breeding Show, Individual Records, Pedigree Charts. $69.95. GENEALOGY SOFTWARE, POB 1151, PORT HURON, MI 48061, (519) 344-3990.

FREE SOFTWARE CATALOG!
Call Toll-Free 1-800-554-1162, Tevex, Inc. Save ⅓ off retail prices. We carry SSI, Elect. Arts, Infocom, and many more!

COMMODORE: TRY BEFORE YOU BUY. Top 25 best-selling games, utilities, new releases. Visa, MasterCard. Free brochure. Rent-A-Disk, 908 9th Ave., Huntington, WV 25701 (304) 522-1665

DISCOUNT SOFTWARE: Amiga/Apple/Atari/C64-128/IBM PC-PCjr/TRS-80/Timex/Sinclair. Free Catalog: WMJ DATA SYSTEMS, 4 Butterfly Dr., Hauppauge, NY 11788

3 GAMES-PHOBOS-Destroy a falling moon. 10101 - Play 21 against your computer. STALACTITE ATTACK-Try to survive in a cave with attacking stalactites-requires joystick, ALL THREE for 32K ATARI, ON DISK, $5 check or M.O. to: M. A. Grossman, W205 N6844 Woodward Ct., Menomonee Falls, WI 53051

Dr. T's MUSIC SOFTWARE
A music composition system with true word processing capabilities for APPLE and COMMODORE computers. Also algorithmic composition programs for computer generation of sequences. Fully editable Bach, Keyboard, Bass and Drum disks. Patch librarians available for Yamaha and Casio synthesizers. Reviewed in Jan '85 COMPUTE! (617) 244-6954, Dr. T's, Dept. C, 66 LOUISE RD., CHESTNUT HILL, MA 02167

DATA BASE 130 FOR THE ATARI 130 XE
Data files up to 64K. Sort on any field. Print address labels or listings. $25. Bill Von Sennet, P.O. Box 14585, Pittsburgh, PA 15234. Specify 810 or 1050.

IBM PCjr OWNERS: We have the software you want. We also have complete 2nd drive additions or kits. FREE CATALOG. OOWL SOFTWARE, 1435 Dundey Sq. N., Columbus, OH 43229

FREE SOFTWARE for C64, C128, IBM & CPM. For info send large stamped (39¢) return envelope to: PUBLIC DOMAIN USERS GROUP, PO Box 1442-A1, Orange Park, FL 32067

TI-99/4A QUALITY SOFTWARE for Business, Home and Entertainment * * BONUS Software Offer! * * Send for FREE catalog to MICRO - BIZ HAWAII, BOX 1108, Pearl City, HI 96782

TAX SPREADSHEET FOR C64, ATARI, TI: $19.95. + $2 s/h. Includes 1040, 2106, 2441, A,B,C,D,E,G,SE,W,1040A. Yearly updates $10. TI-99/4A w/ExtBasic, Atari w/48K or more. Specify disk or tape. Steve Karasek, 855 Diversey, St. Louis, MO 63126 (314) 961-2052

ATTENTION TI99/4A OWNERS !!!
See our ad in Product Mart.
The 99/4A National Assistance Group
(305) 583-0467

**** ATARI USERS ** QUEST FOR LIFE!!**
Realistic Adventure with weapons, killers and traps. $14.95 tape/disk. FREE CATALOG. Octavian Software, Box 174, Republic, OH 44867

GAMES FOR C-64!...ANY CHOICE $19.95 T/D. Navy War, Startrek, Football, Risk. Free game w/each buy. RK Games, 420 W. Central, Medicine Lodge, KS 67104. (316) 886-5097

MISCELLANEOUS

HELP IS ON THE WAY!
Just call 1-800-334-0868 to get your free copy of the latest COMPUTE! Books Catalog! If you need help in getting information on all of the latest COMPUTE! book titles available plus all COMPUTE! backlist titles, call us today!

Maxell MD1, $1.29-MD2, $1.99. Dysan 104/1D, $1.79-104/2D, $2.39. Shipping $3.75. Also Verbatim, IBM, 3M, BASF. TAPE WORLD, 220 Spring St., Butler, PA 16001, 1-800-245-6000. Visa, MC.

EARN MONEY, PART OR FULL TIME, AT HOME with your computer-manual & forms-$9.95. Write Computer Programs for Profit! How-to guide with forms, letters, tips-$7.95. Also-Computer Consultant Handbook. How to earn $ consulting with business-$7.95. JV Tech, P.O. Box 563, Ludington, MI 49431

Discount computer printer ribbons for all makes/models Ex: Epson 1500 Nylon $6.99. Catalog: TWS 1314 4th Ave., Coraopolis, PA 15108 (412) 262-1482 Visa or MasterCard.

NEED THAT CIRCUIT DESIGNED into artwork, or that artwork fabbed into circuit boards, and the PCB's assembled? Then send requirements for free quote to the PC Master, 10221 Slater Ave., Suite 103-329, Fountain Valley, CA 92708

1985 INDEX TO ALL COMMODORE MAGAZINES. C64/128 disk with search/browse/print of articles, programs, reviews. Send $16.95 ck or m.o. to: PCdex, Box 563, Dayton, OH 45409

IBM PCjr REPORT: THE NATIONAL NEWSLETTER. PCjr-specific articles, reviews, Public Domain from across the nation. $18./yr. PCjr CLUB, POB 95067, Schaumberg, IL 60195

RIBBONS for MOST PRINTERS at LOW PRICES!
DELTA MICRONICS
BOX 10933, ERIE, PA 16514-0933
(814) 455-5667

ROBOT KITS - The MOVIT family of computer robots that you have been reading about.
PEPPY - navigates by two-way sound sensor, zips around on three wheels. $24.99 + $2 s/h.
AVOIDER - infra-red obstacle avoidance system,, walks on six legs, $44.99 + $2 s/h.
MR. BOOTSMAN - Wired remote control six leg robot. Two motor control, $34.99 + $2 s/h.
Catalog $1.00. Send orders to: CompuTek, 530 South 6th St., Sunnyside, WA 98944

COMPUTE! Classified is a low-cost way to tell over 350,000 microcomputer owners about your product or service.

Rates: $25 per line, minimum of four lines. Any or all of the first line set in capital letters at no charge. Add $15 per line for boldface words, or $50 for the entire ad set in boldface (any number of lines.)
Terms: Prepayment is required. Check, money order, American Express, Visa, or MasterCard is accepted. Make checks payable to COMPUTE! Publications.
Form: Ads are subject to publisher's approval and must be either typed or legibly printed. One line equals 40 letters and spaces between words. Please underline words to be set in boldface.
General Information: Advertisers using post office box numbers in their ads must supply permanent address and telephone numbers. Orders will not be acknowledged. Ad will appear in next available issue after receipt.
Closing: 10th of the third month preceding cover date (e.g., June issue closes March 10th). Send order and remittance to: Harry Blair, Classified Manager, COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. To place an ad by phone, call Harry Blair at (919) 275-9809.
Notice: COMPUTE! Publications cannot be responsible for offers or claims of advertisers, but will attempt to screen out misleading or questionable copy.