Joseph Roehrig
Box 74
Middle Village NY 11379

3-D Tic-Tac-Toe

a winner with the whole family!

Now that you have your computer running, it is time to entertain your family and friends. At the same time, you should impress them with your computer's brilliance. A game is the natural medium to introduce others to your new sophisticated toy and a familiar game is a wise choice. Tic-tac-toe is very well-known and a logical choice for your demonstration.

IBM had a tic-tac-toe game in its pavilion at the 1964 New York World's Fair. IBM's game could never lose, but also could never win against a knowledgeable player due to the simplicity of the game. A standard tic-tac-toe game has a two-dimensional 3-box by 3-box game board. There are only nine possible moves, making the game rather easy to play for both man and machine.

To improve our display game, the board has been expanded from the standard 3 by 3 to a 4 by 4. This adds to the complexity of the game, but a skilled player can still stand off a computer. Here we add a third dimension and increase the size of the board. Now the game becomes a real challenge with its 4 by 4 by 4 cubic look.

Three Versions

This article describes and provides programming details for three versions of tic-tac-toe. All three versions will be derived from one relatively short sixty line BASIC language source listing. The program is written in Altair 3.2 BASIC and is geared for a video terminal with 80 characters and 24 lines. (An option to the coding is described to reduce the printed output if desired.)

The versions are:

Version 1 — a simple 4 by 4 with quick response time.

Version 2 — a 4 by 4 by 4 which can be beaten and respond to each move in 25 seconds or less.

Version 3 — an almost unbeatable 4 by 4 by 4 game. I'm forced to say almost because the possible moves can go as high as 64 factorial. An easier way of pointing out the numerous possibilities is to say that the first two moves taken by each player can take any one of 15,249,025 combinations.

Fig. 1 shows what the computer prints out for all three verions. Boards 2, 3 and 4 are not needed in the two-dimensional 4 by 4 Version 1. Each board has sixteen possible positions. To move, you choose the board (1 to 4) and the position (1 to 16). The sixteen position numbers are the same for each of the four boards.

Version 3 — The Rough One!

An easy way to proceed is to describe Version 3, the most complex, and then discuss the modifications necessary to use the other two. Fig. 2. is the flowchart of the game. First, the computer sets all of its 64 board squares to double blanks. During the game, the computer's squares will be marked CC and the player's UU. The board is different from Fig. 1. The computer really has only one board that contains 64 squares. Fig. 3 shows how the computer's 64 squares correspond to the player's four boards of sixteen squares each. Also detailed in Fig. 3 are the 76

```
YOUR MOVES ARE UU AND I'M CC
POSITION # ARE
                                  1   5   9  13
                                  2   6  10  14
                                  3   7  11  15
                                  4   8  12  16

  BOARD 1        BOARD 2        BOARD 3        BOARD 4

  UU/ / /          / / /          / / /          / / /
  ///////////    ///////////    ///////////    ///////////
    / / /        CC/ / /          / / /          / / /
  ///////////    ///////////    ///////////    ///////////
    / / /          / / /          / / /          / / /
  ///////////    ///////////    ///////////    ///////////
    / / /          / / /          / / /          / / /

     YOUR BOARD, POSITION? 1,4
     I WANT BOARD 1 POSITION 2

     (EVERYTHING EXCEPT LINE 1 IS REPRINTED UP TO
     ''YOUR BOARD ...'')
```

Fig. 1. Display of game being run.

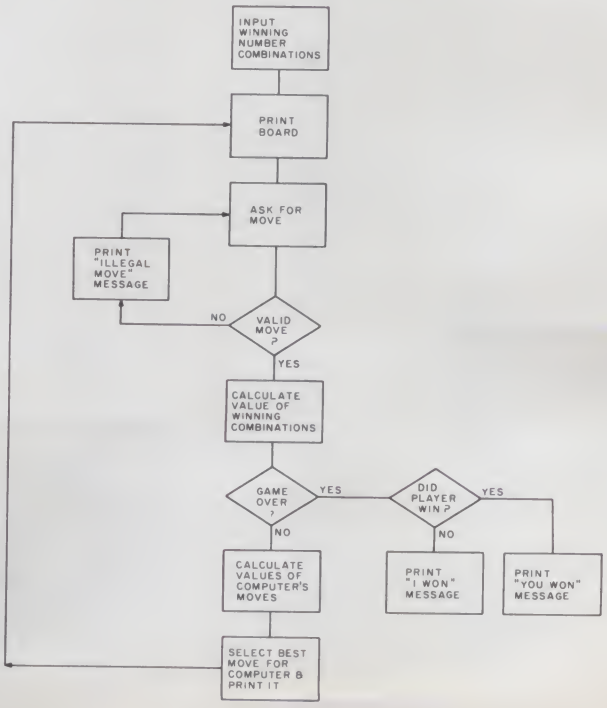

Fig. 2. Tic-tac-toe flowchart.

possible winning combinations that must be stored in the computer's memory. Only winning moves 1 to 10 are needed for the two-dimensional game. Wins 1 to 40 have moves that are all on one board. The winning moves described as 41 to 76 have one move on each of the four boards. Some of these are tricky and a study of Fig. 3 will familiarize you with all of the possible ways of winning.

The computer then asks for your move (Fig. 1). Your move is a board number, a comma, and a position number. These two figures are converted into a computer position which is described in (Fig. 3). If you move to an occupied space or type in an invalid move (not a number from 1 to 4 followed by a comma and a number from 1 to 16), the computer will again ask for your move. Note: in the two-dimensional game, a move to any board other than board 1 will result in a lost move.

The computer now calculates the value of each of the 76 possible winning combinations. The value is equal to the sum of the values assigned to each of the four squares or board positions contained in the winning combination. The values of the board positions are:

0 for an unoccupied box — prints 2 blanks on game board.

1 for your boxes — prints UU on appropriate game board position.

5 for computer occupied boxes — prints CC on game board.

These values are important and are used in all move decisions.

The computer now sees if the value 4 exists in any of the 76 win possibilities. If 4 exists, you have beaten the computer and the game is over. A four designates a player's win, since the only way four can exist is to have a 1, a player's box, in each of

```
10 DIM S(64), W(3,76), S$(64), V(76)
20 FOR A = 1 TO 10 : FOR A1 = 0 TO 3 : READ W(A1,A) : NEXT A1,A
30 FOR A = 1 TO 3 : A1*10 : FOR A2= 1 TO 10 : FOR A3 = 0 TO 3
40 W(A3,A1+A2)=W(A3,A2)+(16*A) : NEXT A3,A2,A
50 FOR A = 41 TO 56: FOR A1 = 0 TO 3
60 W(A1,A) = (A1*16)+A-40 : NEXT A1,A
70 FOR A = 57 TO 76 : FOR A1 = 0 TO 3 : READ W(A1,A) : NEXT A1,A
72 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,1,5,9,13,2,6,10,14
74 DATA 3,7,11,15,4,8,12,16,1,6,11,16,4,7,10,13,1,22,43,64
76 DATA 5,22,39,56,9,26,43,60,13,26,39,52,2,22,42,62,14,26,38,50
78 DATA 3,23,43,63,15,27,39,51,4,23,42,61,8,23,38,53,12,27,42,57,16,27,38,49
80 FOR A = 1 TO 64: S$(A)=" " : S(A)=0 : NEXT A
83 DATA 1,21,41,61,1,18,35,52,4,19,34,49,4,24,44,64
84 DATA 13,25,37,49,13,30,47,64,16,31,46,61,16,28,40,52
85 PRINT"YOUR MOVES ARE UU AND I'M CC"
90 GOSUB 1000 : "INPUT "YOUR BOARD, POSTION";A1,A2
100 A=((A1-1)*16)+A2
105 IF A >64 OR >1 A THEN PRINT"ILLEGAL MOVE" : GOTO 90
110 IF S(A) <>0 THEN PRINT"YOU CAN'T MOVE THERE" : GOTO 90
120 S(A)=1 : S$(A)="UU"
190 M5=0 : FOR A = 1 TO 76
192 A2= W(0,A) : A3=W(1,A) : A4= W(2,A) : A5=W(3,A)
194 V(A)=S(A2)+S(A3)+S(A4)+S(A5)
196 IF V(A)=4 THEN 410
198 IF V(A) = 15 THEN M5=A
199 NEXT A : IF M5 <>0 THEN 365
200 M3=9
204 Y1=0
205 FOR A = 1 TO 64 : M2=0
210 IF S(A) <> 0 THEN 350
215 Y1 = Y1+1
220 FOR A1 = 1 TO 76
225 FOR A2 = 0 TO 3 : IF A=W(A2,A1) THEN 230
228 NEXT A2: GOTO 300
230 A6= V(A1)
260 IF A6=3 THEN M4=A : GOTO 390
270 IF A6 = 0 THEN 300
280 IF 5 >A6 THEN M2 = M2 + A6 A6 : GOTO 300
290 A7= INT(A6/5) : IF A7 = A6/5 THE M2=M2+A7
300 NEXT A1
320 IF M2 >M3 THEN M3=M† : M4=A
350 NEXT A : GOTO 390
365 FOR A1 = 0 TO 3 : A6 = W(A1,M5) : IF S(A6) = 0 THEN M5=A6 : GOTO 368
367 NEXT A1
368 PRINT "THE OLD"; M5
370 S$(M5)="CC" : A1= INT(M5-1/16)+1 : A2= M5-((A1-1)*16)
380 PRINT"I WON WITH BOARD";A1;"POSITION";A2 : GOSUB 1000
382 INPUT"READY";A1 : GOTO 80
390 S$(M4)="CC" : S(M4)=5
392 A1=INT((M4-1)/16) : A2=M4-((A1-1)*16)
400 PRINT"I WANT BOARD";A1;"POSITION";A2 : GOTO 90
410 PRINT : PRINT"YOU WON" : GOSUB 1000 : GOTO 80
1000 PRINT"POSITIONS ARE"; : FOR A + 0 TO 3 : FOR A1=1 TO 13 STEP 4 : A2=20 + (A1*4)
1100 PRINT TAB(A2); A+A1 ; : NEXT A1 : PRINT : NEXT A :PRINT:PRINT
1105 FOR A=0 TO 3 : PRINT TAB(A*15);"BOARD";A+1; : NEXT : PRINT
1107 PRINT : PRINT
1110 FOR A = 1 TO 4 : A1 = 0 TO 48 STEP 16 : A2 = A +A1
1120 PRINT S$(A2);"/";S$(A2+4);"/";S(A2+8);"/";S(A2+12);" ";: NEXT A1
1125 IF A=4 THEN 1130
1127 PRINT : FOR A2 = 1 TO 4 : PRINT"////////// "; : NEXT A2
1130 PRINT : NEXT A : PRINT : PRINT : RETURN
```

Program A. BASIC program for Verson 3 of three-dimensional tic-tac-toe.

```
190 M5=0 : Q=0 : FOR A = 1 TO 76
197 IF V(A)=3 THEN Q=A
201 IF Q=0 THEN 205
202 FOR A9= 0 TO 3 : A6 =W(A9,Q) : IF S(A6)=0 THEN M4=A6 : GOTO 390
203 NEXT A9
220 FOR A1= 60 TO 76 STEP 2
280 IF 5 >A6 THEN M2=M2+A6 : GOTO 300
350 NEXT A : IF M3 <>0 THEN 390
352 FOR A =1 TO 64 : IF S(A)=0 THEN M4=A : GOTO 390
354 NEXT A
```

Note: In line 280, change M2=M2+A6 to M2=M2+A6↑A6 to make this version harder to beat.

Program B. Modifications necessary to obtain Version 2 of the game.

Computer Boards:

Board #1	Board #2	Board #3	Board #4
1 5 9 13	17 21 25 29	33 37 41 45	49 53 57 61
2 6 10 14	18 22 26 30	34 38 42 46	50 54 58 62
3 7 11 15	19 23 27 31	35 39 43 47	51 55 59 63
4 8 12 16	20 24 28 32	36 40 44 48	52 56 60 64

Winning moves per computer boards:

1)	1 2 3 4	39)	49 54 59 64
2)	5 6 7 8	40)	52 55 58 61
3)	9 10 11 12	41)	1 17 33 49
4)	13 14 15 16	42)	2 18 34 50
5)	1 5 9 13	43)	3 19 35 51
6)	2 6 10 14	44)	4 20 36 52
7)	3 7 11 15	45)	5 21 37 53
8)	4 8 12 16	46)	6 22 38 54
9)	1 6 11 16	47)	7 23 39 55
10)	4 7 10 13	48)	8 24 40 56
11)	17 18 19 20	49)	9 25 41 57
12)	21 22 23 24	50)	10 26 42 58
13)	25 26 27 28	51)	11 27 43 59
14)	29 30 31 32	52)	12 28 44 60
15)	17 21 25 29	53)	14 30 46 62
16)	18 22 26 30	54)	13 29 45 61
17)	19 23 27 31	55)	15 31 47 63
18)	20 24 28 32	56)	16 32 48 64
19)	17 22 27 32	57)	1 22 43 64
20)	20 23 26 29	58)	5 22 39 56
21)	33 34 35 36	59)	9 26 43 60
22)	37 38 39 40	60)	13 26 39 52
23)	41 42 43 44	61)	2 22 42 62
24)	45 46 47 48	62)	14 26 38 50
25)	33 37 41 45	63)	3 23 43 63
26)	34 38 42 46	64)	15 27 39 51
27)	35 39 43 47	65)	4 23 42 61
28)	36 40 44 48	66)	8 23 38 53
29)	33 38 43 48	67)	12 27 42 57
30)	36 39 42 45	68)	16 27 38 49
31)	49 50 51 52	69)	1 21 41 61
32)	53 54 55 56	70)	1 18 35 52
33)	57 58 59 60	71)	4 19 34 49
34)	61 62 63 64	72)	4 24 44 64
35)	49 53 57 61	73)	13 25 37 49
36)	50 54 58 62	74)	13 30 47 64
37)	51 55 59 63	75)	16 31 46 61
38)	52 56 60 64	76)	16 28 40 52

Fig. 3. Computer's board and winning moves.

the boxes making up a winning combination.

If this condition does not exist, the program continues. The computer now checks to see if a 15 occurred during the previous evaluation. If it exists, the computer wins on this move. Fifteen — not twenty — is a winner, since unlike the player, the computer has not selected its move. Thus 15 means the computer has three boxes in a winning combination that has one unoccupied square. All the computer must do to win is find the box in the combination that equals zero and then designate that box as its move. Naturally, sixteen represents a block by the player. If the computer does not win, we continue and the computer selects its move.

The evaluation of the computer's move considers each of the sixty-four possible boxes that are unoccupied. The computer checks every unoccupied box to see which of the 76 winning moves contains that box. For each winning move containing the box or square under consideration, points are given to the box's evaluation (see Table 1).

Only five values of the winning combinations are of any importance in the evaluation process and a 3 causes an automatic move to block. The block is always taken, since it has already been established that the computer cannot win on this particular move and an unblocked 3 gives the player the opportunity of winning on the next move. Therefore, the 3 must be blocked. The values labeled "all others" are 6, 7, 11, etc. They have no value, since both a UU and a CC are already found in the particular winning combination. Thus, neither the player nor the computer can win with the combination in question.

A winning combination having two of the player's moves in it and two unoccupied squares is awarded four points. This is more than a combination having two of the computer's moves in it and two unoccupied boxes, which is awarded only two points. This is done because the player always moves ahead of the computer, forcing the computer to play defense more than offense. This priority also avoids the situation shown in Fig. 4 — the strategy that can beat the computer if this priority is not employed.

All of the possible points are totaled for every winning combination involving the box under evaluation. The box receiving the highest point score becomes the computer's move.

That completes a description of Version 3 of the tic-tac-toe game which I feel cannot be beaten by a human opponent.

Now for the bad news; it

To beat the computer, the player should establish a situation where three of his boxes are not strung together until there are at least two opportunities established by the string of three. The player establishes the UUs and then the XX making it impossible for the computer to block both winning combinations.

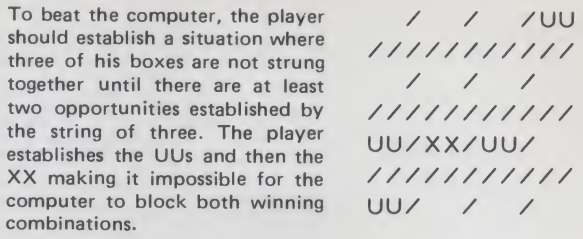

Fig. 4. Winning strategy.

Value of Winning Combination	Points Awarded to Box
1	1
2	4
3	no points, moves to block
5	1
10	2
all others	0

Table 1.

Version 2 Game

Player's Move	Computer's Move
1,4	2,8
1,13	2,10
1,11	1,14
1,3	3,6
1,7	4,2
Computer Wins	

Version 2 Game

Player's Move	Computer's Move
1,4	2,8
1,13	2,10
1,11	1,14
1,3	3,6
4,2	1,8
1,7	1,10
1,15	
Player Wins	

Version 3 Game

Player's Move	Computer's Move
1,4	1,1
1,13	1,7
2,4	4,4
2,13	4,13
2,11	1,16
2,3	2,7
3,4	4,1
3,13	2,1
3,1	3,3
3,14	2,2
Computer Wins	

Fig. 5. Sample games.

takes about three minutes for the computer to calculate its first move. The time required to select the computer's move gradually goes down as the game progresses.

Version 2 — Can Be Beaten

Version 2 is also three-dimensional, however, it moves in 25 seconds or less and you can beat it. The game is played exactly like Version 3, except: a) All of the combinations are not evaluated. Therefore, the computer does not always pick its best move and it is much faster. b) The computer plays offense equal to defense and does not always prevent the player's winning strategy (Fig. 4).

This version is the one I recommend that you use. Fig. 5 shows the moves from a few sample games. In the first example the player loses to Version 2; next the player beats Version 2; and last the winning moves are pitted against Version 3 without any success for the player. Depending on the skill of the player, Version 2 can be difficult to beat. In fact, it can only be beaten by the strategy described in Fig. 4. After a friend has lost a game or two to Version 2, you can then take over and easily conquer the computer much to your friend's surprise.

On all computer wins, the computer prints the old XX. XX represents the winning combination as detailed in Fig. 3.

The first version of the game is two-dimensional and needs no explanation; it is played like the other two. However, as mentioned earlier, do not move to boards 2, 3, or 4, or your turn will be lost.

The Program Listings — And Modifications

Program A is the source listing for Version 3. Program B lists the additions, deletions and changes necessary for Version 2. Only winning combinations 60, 62, 64, 66, 68, 70, 72 are checked in Version 2. By changing line number 220 you can change the combinations (and the number of combinations) used during the execution of the program. I suggest always using combinations above 40 in order to give the computer a better chance of winning. These combinations all use the third dimension, therefore making the human player's defense more difficult.

To play the two-dimensional game, Version 1, changes shown in Program C must be incorporated into Program A.

As mentioned earlier, all versions of the game are designed for use on a video terminal and the games display the boards after every two moves (one by the player and one by the computer). They also print the computer's moves. Therefore, to avoid the board printings only, make the following change: 1000 RETURN.

You will now have to keep track of the boards on your own, unless you have a memory equal to that of the computer.

If you enter the program into your computer, you will probably make typographical errors. The two lines in Program D will help you determine whether the proper winning combinations are entered. These lines will be the first inputs you encounter.

In order to rejoin a game at any given point during debugging or for other reasons, use the lines in Program E.

These lines allow you to enter all player and computer positions. First, enter all player positions into the computer using the computer's format (numbers 1 to 64) and a 0 to stop. The same procedure is then repeated for the computer.

The program was run with 12192 words of memory. Including the BASIC, about 2500 words are left for your enhancements.

Good luck, and if you beat my almost unbeatable game, change line 290 as shown in Example 4. This will make the computer play even more defensively. I have not yet beaten Version 3, so I have not tried this modification. Figs. 6 and 7 will be an aid in debugging and enhancing the program, since they describe the major variables and computer functions by line number. ■

S(64) — value of all boxes.
S$(64) — board character for each box.
W(3,76) — 76 winning combinations.
V(76) — value of winning combinations.
M5 — if positive, possible computer win if loss has not occurred.
M2 — accumulates value of all combinations that involve the box being evaluated.
M3 — highest value for a box thus far.
M4 — number of the box having the above highest value.
A6 — value of combination being examined.

Fig. 6. Major variables.

10 — dimension variables.
20 to 84 — read winning combinations as data or calculate them from existing winning combinations.
80 to 120 — player's move.
190 to 199 — evaluate winning combinations and check for computer loss or possible win.
200 to 350 — select computer's move.
365 to 410 — print move information.
1000 to 1130 — subroutine to print board.

Fig. 7. Functions by line number.

```
190 M5=0 : FOR A = 1 TO 10
205 FOR A = 1 TO 16 : M2=0
220 FOR A1 = 1 TO 10
```

Program C. Changes to obtain Version 1.

```
81 INPUT A,B : FOR C= A TO B : FOR D = 0 TO 3 : PRINT W(D,C);: NEXT : PRINT: NEXT C
82 INPUT Q : IF Q>1 THEN 81
```

Program D. Entry program for Winning Combinations data.

```
81 INPUT Q : IF Q<>0 THEN S(Q)=1 : S$(Q)="UU" : GOTO 81
82 INPUT Q : IF Q = 0 THEN S(Q)=5 : S$(Q)="CC" : GOTO 82
```

Program E. Modifications necessary to have "interrupted" game.

```
290 A7=INT(A6/5) : IF A7+ A6/5 THEN M2=M2+A7-1
```

Program F. Statement to obtain the ultimate unbeatable game.