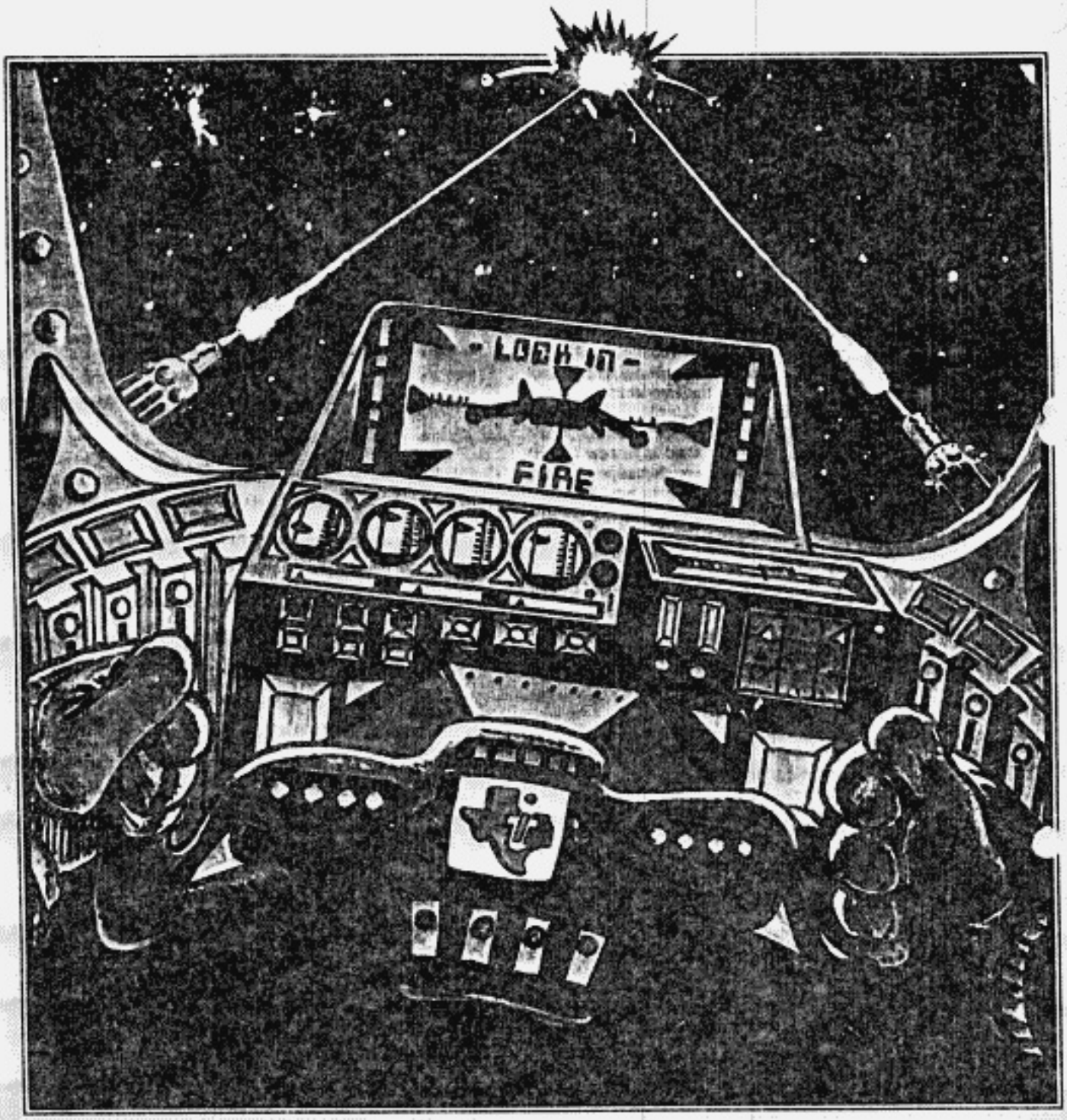
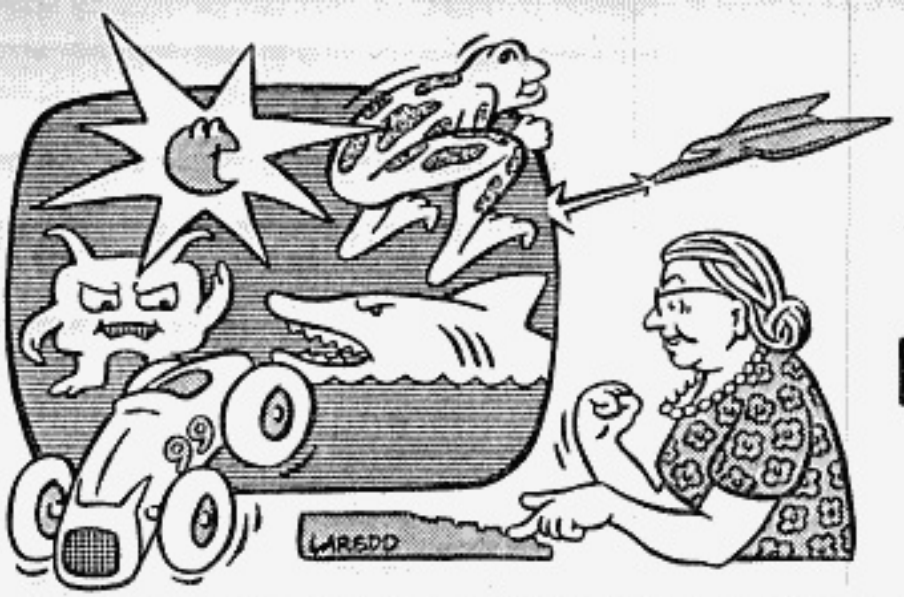
7
Computer Gaming



7

Computer Gaming

Action-packed games to delight arcade fans and mind-stretching challenges for strategists.



The JOY of COMPUTER GAMING

It's A Dirty Job, But Somebody's Gotta Do It . . .

Over the last couple of months, I've had virtually no rest. First it was those pesky aliens: They hurled bombs, missiles, mines, and laser blasts at me around the clock. Some even tried to gobble me up on sight! No respect at all. . . These hordes of menacing foes must have come from nearly a dozen different hostile worlds. (Why is it I've never seen a "friendly" alien?) Each of these worlds evidently has its own individual concept of combat strategy, weapon design, ethics and morality because the modes and severity of attacks differed widely. One thing, however, that all these dastardly devils had in common was their quarry—me!

Some of the attacking hordes were accompanied by a malevolent thumping as their precise marching formation advanced hypnotically toward my flimsy barricade. Others stayed stationary but hurled down torrents of lethal missiles that I had to alternately duck and target my lasers against. (My neighbors must have been really surprised when they noticed all that debris strewn across their yards. . .) What? Was I nervous? Not too—that is, not until I had to pilot all those strange land and space craft—everything from X-wing fighters to futuristic prairie schooners. Just when I'd feel comfortable at the controls of one, Ka-boom!—I'd be under vicious attack, or c-r-r-unch!—smack in the middle of a deadly asteroid belt. Nothing like huge chunks of space rocks whizzing around your head to keep you on your toes. . .

But it didn't end with all those downright nasty aliens and slimy, vile space creatures. Oh no, not by a long shot. There were still the Empire forces to contend with. Here, however, the battles were more scattered and slower paced; I had time to launch torpedoes and probes, as well as assess casualty reports and plan long-term strategy. Just when everything was going so well, a Red Alert brought me back down to earth. It seems the Cold War was no longer so cold. . .and my country needed me to command a SAM (surface-to-air missile) site. With all that sophisticated RADAR equipment, it shouldn't have taken too long to finish "locking in" on the enemy missiles and blasting them to smithereens, but as fate would have it, another series of emergencies sent me packing—first to rescue a downed helicopter crew from shark-infested waters, and then on a hazardous journey to the moon and Mars, where I had to jockey my landing craft over some pretty rough terrain.

You'd think by then I should have received a few words of thanks, wouldn't you? But no, the moment I landed on Mars, some terrorists decided to have a field day. . .and there I was right back in the thick of things—commanding a bomb squad. Now defusing a time bomb is no Sunday picnic! If my nerves could withstand that heart-thumping activity, you'd think I'd be in pretty good shape for a few more adventures yet to come.

But nothing prepared me for what I was up against next. Certainly, no one told me when I took this job that hundreds of horrible deaths awaited just around the corner. All they talked about was the treasure and glory! But when I reached the edge of the high cliff, it was already too late to turn back: On my left, a hungry python slithered toward me; to the right, a quicksand bog surrounded by bleached bones awaited; and behind me, a large grizzly bear blocked my only path into the forest.

Well, luckily I got out of that one with my skin, but one adventure led to another. . . And before I could take some well-earned rest, I somehow had gotten involved in a pirate treasure hunt, an escape from an ancient pyramid, the ferreting out of an awesome secret on a savage island, saving a Count trapped by a fiendish curse, preventing a nuclear reactor from blowing its top, and alternately exploring a ghost town, mysterious fun house, ancient alien civilization, enchanted treasure world, and a dark kingdom populated by orcs, dwarves, an old dungeon master, a beautiful princess in distress, and an evil ringwraith.

Whew—I never thought that one mortal could get so tired. What I really needed was a chance to relax and unwind. . . So, handing my ticket to adventure over to Indiana Jones, I planned on doing nothing but sitting back in my favorite easy chair and listening to some good music. But They had other plans for me. It was no use complaining; I'd heard the argument a hundred times before: "It comes with the territory. . ." For some reason or other I was needed to run up a bankroll by betting on the ponies, to lead a championship baseball team to victory, to bring back a

shiny bowling trophy, to don my skis to better the old slalom record, to outrace a suicide car, and then to take part in a grueling decathlon.

After somehow getting through that long, long decathlon, I sat down to a nice big bowl of Wheaties and planned my R&R. Nothing too strenuous. . .nothing too mentally demanding. . .just some good clean fun. So off to the casinos for some baccarat, blackjack, craps, poker, and the slots. It was fun while it lasted, but They needed me back on the job again.

I knew something really BIG must have been in the works because of the way my training for the forthcoming mission was being carried out: plenty of practice with challenging word games, concentration exercises and contortions with cantankerous colored cubes. They obviously wanted my wits sharp for the BIG assignment coming up. But before I'd find out what it was, there was an obstacle course to negotiate, and then the final test of my state of mental readiness—passage through a series of simply complex, complexly simple 3-D mazes. I almost didn't make it through that one. . .

Now I was ready. The BIG assignment finally came in: Someone was needed to guide a dumb chicken safely across a 20-lane highway. . . What? Enough is enough! Tell 'em I'm not here. Guide a chicken across a road like that? Instant chicken salad—with me probably ending up being accused of fowl play. . . Let 'em get somebody else for that one. I wouldn't do it now even if They awarded me the Pullet-ser Prize!

Micro Motivation Comes Full Circle

I've been sitting here now several hours thinking and wondering—thinking about those psychedelic-sounding escapades of mine, and wondering about the fantastic powers of imagination that we all must have—letting us see what we want or expect to see. In my case, it was easy because I had a partner—one who was, incidentally, a lot more patient than that dumb chicken I eventually got teamed up with. Who was this patient partner? Some gaunt guru? Or sinewy sorcerer? No, none of these. My partner in all this was a friendly Texan—a TI Home Computer equipped with the latest in games software.

I have never really cared very much for games. And even when it became obvious that microcomputers were rapidly becoming the ultimate "games machines," I still felt that all the excitement of video games was just a passing fancy. It was my belief that the popularity of computer games was simply due to people just trying to find additional uses for machines that they bought primarily for other purposes. Now I know better. . .

What we're now just starting to see happen is actually the reverse: This year, several million consumers will be considering interactive video games—as opposed to passive TV watching—that they can play at home in the company of friends and family, instead of plunking their quarters down coin slots at crowded arcades or all-night grocery stores. When they start to shop around and compare prices and features, increasing numbers of them will start to find that the stand-alone, cartridge-based, dedicated games machines can be almost as much money as the new breed of lower-cost microcomputers.

The handwriting is already on the wall: As the price of microcomputers falls even lower, many, many more consumers who were initially looking for video games machines will be able to justify the slightly higher cost of a full computer on the basis of potentially being able to do so much more than "just play games." Ironic, isn't it. . .

A few years ago, the great topic of speculation and cause for disagreement in the microcomputer industry was how to best increase public awareness and acceptance of these miracle machines so that a mass market with its lofty goal of "a computer in every home" could be eventually realized. Everything from electronic mail and banking, to education, home management, and tax/financial record keeping was nominated as being a likely candidate for the magic catalyst. Sure, entertainment was mentioned, but it was usually lumped together somewhat amorphously with home management and education. Nowhere do I remember anyone coming out and stating that it would be computer gaming that would ultimately be this catalyst and pave the way.

The Seriousness of Playing Games

But regardless of whether video games are a primary or secondary motivation for getting a microcomputer, it's rapidly becoming obvious that electronic game playing isn't all just a game. Psychiatrists, psychologists, therapists and educators are discovering how video games can dramatically benefit their players. We hear reports of how the games are speeding eye-hand coordination, sharpening driving and math skills (since the intricate strategies and geometric patterns of many video games provide painless instruction in logic, trigonometry and physics), preventing youth from being stricken by technological "future shock," and providing an emotional rescue (by dissipating anger and frustration, assuaging loneliness and allowing both the recapture of lost athletic prowess as well as the prowess that never was).

Application of video game playing as a form of therapy is definitely on the rise. We're now seeing this technique used in treating brain-damaged victims of strokes, accidents and senile dementia. The most impressive results, however,

have come from work with retarded or emotionally disturbed children. Here, video games often break through where other methods fail. Psychologists have credited this to the "mastery experience" that is now possible for children who formerly were not able to be good in anything else. Until their exposure to games, they have never had a refuge of accomplishment from which to deal with the outside world. Once children become good at something (and, as a result, proud of their achievements), their attitudes and performance in other activities also dramatically improve.

The Hardware Sets the Stage

As opposed to video games machines that are designed to be just "machines that play games," microcomputers are usually designed to perform many types of jobs, or handle certain types of work more efficiently. This architectural design determines the gaming environment that a particular computer will present to its users. Any limitations or constraints will be very obvious. For example, if a computer was designed without color graphics capability, then the games software compatible with this particular machine could not utilize color. Likewise, sound effects, music, 3-D animation and synthetic speech are other game-enrichment capabilities that a microcomputer may or may not possess.

A comparison of all presently available microcomputers yields the surprising conclusion that only the Texas Instruments TI-99/4A personal computer has all the above named capabilities and permits programmers to use them all in the same program. This represents an abundance of "raw materials" from which to construct games. When you combine this with TI's fast and powerful 16-bit microprocessor (TMS9900), it becomes apparent that these Texas Instruments personal computers offer one of the best gaming environments available.

The separate Video Display Processor chip (TMS9918A) inside the TI-99/4A is a good example of TI implementing internally in hardware what other computers require programmers to do in software (if indeed it can be done at all). This very sophisticated device gives the games programmer the ability to simply access and set in motion (independently of the program logic) 32 smoothly moving colored objects called "sprites"—objects whose shapes can very easily be defined, magnified, colored, given a 3-D overlapping appearance and checked for collisions. These are the modular components from which many more exciting arcade-type games will be constructed in the future.

ANTI-AIRCRAFT GUN

TI BASIC

Despite the fact that action games programmed in a high-level language such as TI BASIC run much slower than in low-level languages such as 9900 Assembly Language or GPL (the programming language of TI's Command Cartridges), it is indeed possible to create reasonably fast "real-time" games if you observe a few rules:

1. Keep the number of moving objects to a minimum.
2. Keep all unnecessary statements out of loops used to move objects.
3. Use only one character to define objects you want to move quickly.
4. Increment the positions by two spaces each loop. This makes the movement slightly jerky, but contributes greatly to the illusion of speed.

I've followed these rules in writing *Anti-Aircraft Gun*. The basic idea of the game is simple: You must shoot down an attacking plane with your missile launcher before it blasts you twice with its laser. The plane attacks at random heights from both the left and right sides. Its speed and frequency of laser fire are dependent on the skill level you choose. Your missile launcher can move along the ground, and even hide behind a barrier; but when it fires a missile, it is committed to its last position until the missed shot passes off the screen. You'll have to move around as much as possible because the plane "remembers" your last position and there is higher probability it will fire the laser near that position. And don't expect too much protection from the barrier: After five laser blasts (or less, if you launch missiles through it), it will disintegrate and leave you exposed.

EXPLANATION OF THE PROGRAM
Anti-Aircraft Gun

Line Nos.	
100-670	Instructions.
680-810	Sets up levels of difficulty.
820-870	Sets up variables to make plane fire more as difficulty increases.
880-1110	Character definitions and color assignments.
1120-1170	Initial displaying of tank.
1180-1220	Displays ground.
1230-1260	Calculates plane's height.
1270-1380	Determines the direction of the plane.
1390-1450	Reads keyboard, and branches to subroutines.
1460-1530	Fires tank's rocket.
1540-1610	Moves plane.
1620-1650	Decides whether plane will fire or not.
1660-1760	Fires plane's laser; checks for hits.
1770-1780	Checks for a hit on the plane by the tank's rocket.
1790-1890	Checks for plane at the edge of the screen.
1900-2030	Determines new direction for the plane.
2040-2090	If plane and tank hit simultaneously, the tank wins.
2100-2180	Determines new direction for the plane.
2190-2330	Calculates score.
2340-2420	Prints score.
2430-2460	Plane is destroyed by the tank.
2470-2620	Calculates if a free game was won; starts over.
2630-2750	Moves tank left.
2760-2880	Moves tank right.
2890	END.

```
100 REM  *********************
110 REM  *                   *
120 REM  *  ANTI-AIRCRAFT GUN *
130 REM  *                   *
140 REM  *********************
150 REM
160 REM
170 CALL CLEAR
180 CALL SCREEN(8)
190 REM  * INTRODUCTION *
200 PRINT TAB(3);"PRESS I FOR INSTRUCT
    IONS"
210 PRINT TAB(8);"AND N FOR NONE"::::::
    ::::
220 CALL KEY(0,LP,PL)
230 IF PL=0 THEN 220
240 IF LP=78 THEN 680
250 PRINT TAB(5);"* ANTI-AIRCRAFT GUN
    *"
260 PRINT
270 PRINT "THE OBJECT OF ANTI-AIRCRAFT
    "
280 PRINT "IS TO DESTROY AS MANY PLANE
    S"
290 PRINT "AS POSSIBLE. TO FIRE YOUR "
300 PRINT "MISSILE, PRESS Q. TO MOVE "
310 PRINT "LEFT, PRESS S, AND FOR RIGH
    T"
320 PRINT "PRESS D."
330 PRINT
340 PRINT TAB(9);"* PLANES: *"
350 PRINT
360 PRINT "THE PLANE FIRES A NEWLY-"
370 PRINT "DEVELOPED LASER. THE PLANE"
380 PRINT "MAY COME FROM LEFT OR RIGHT
    "
390 PRINT "WARNING! IT HAS A RADAR THA
    T"
400 PRINT "REMEMBERS YOUR LAST FIRING"
410 PRINT "POSITION & TRIES TO GET YOU
    "
420 PRINT "THERE; BETTER MOVE AROUND."
430 PRINT :::
440 PRINT "PRESS ANYTHING TO CONTINUE"
450 PRINT
460 CALL KEY(0,KL,LK)
470 IF LK=0 THEN 460
480 PRINT TAB(8);"* BARRIER: *"
490 PRINT
500 PRINT "THE LASER CAN'T PENETRATE "
510 PRINT "THE BARRIER, BUT THE BARRIE
    R"
520 PRINT "CAN SUSTAIN ONLY 5 DIRECT "
530 PRINT "HITS. YOU CAN FIRE WHEN BE-
    HIND IT, BUT THIS WILL ONLY
    "
550 PRINT "SHORTEN ITS LIFE."
560 PRINT
570 PRINT TAB(8);"* SCORING: *"
580 PRINT
590 PRINT "PLANES ARE SCORED ACCORDING
    "
600 PRINT "TO HEIGHT: 5 FOR LOWEST, 20
    "
610 PRINT "FOR HIGHEST. IF YOUR SCORE
    "
620 PRINT "IS OVER 100 THEN YOU GET A
    "
630 PRINT "FREE BONUS GAME."
640 PRINT :::::::
650 PRINT "PRESS ANYTHING TO CONTINUE"
660 CALL KEY(0,KL,LK)
670 IF LK=0 THEN 660
680 CALL CLEAR
690 CALL COLOR(2,2,8)
700 PRINT "INPUT LEVEL OF DIFFICULTY:"
710 PRINT
720 PRINT "(1) PRO"
730 PRINT
740 PRINT "(2) INTERMEDIATE"
750 PRINT
760 PRINT "(3) NOVICE"
770 PRINT
780 PRINT "(4) BEGINNER"
790 PRINT :::::::
800 CALL KEY(0,DIF,KS)
810 IF (DIF<49)+(DIF>52)=-1 THEN 800
820 REM  PLANE FIRING MORE  AS DIFFIC
    ULTY INCREASES
830 DIFF=DIF-38
840 MS=-9
850 AA=1
860 ZZ=3
870 CALL CLEAR
880 REM  DEFINE CHARACTERS
890 REM  BAS=BARRIER, LS=  LASER, RS
    =ROCKET, PS & PRS= PLANE, TLS & TR
    S=TANK, DS=  DESTRUCTION
900 BAS="FFFFFFFF"
910 DS="2A0441228994522A"
920 LS="1818181818181818"
930 RS="1818181818183C7E"
940 PS="6030981FFF983060"
950 PRS="060C19FFFF981806"
960 TLS="071F7FFFFF7F1F07"
970 TRS="E0F8FEFFFFFEF8E0"
980 CALL CHAR(120,DS)
990 CALL CHAR(144,LS)
1000 CALL CHAR(97,RS)
1010 CALL CHAR(112,TLS)
1020 CALL CHAR(113,TRS)
1030 CALL CHAR(101,BAS)
1040 CALL CLEAR
1050 BA=0
1060 CALL COLOR(2,16,2)
1070 CALL COLOR(16,2,2)
1080 CALL COLOR(15,12,8)
1090 CALL COLOR(13,3,3)
1100 CALL COLOR(12,7,8)
1110 T=15
1120 REM  INITIAL PRINTING OF TANK
1130 CALL HCHAR(24,T,112)
1140 CALL HCHAR(24,T+1,42)
1150 CALL HCHAR(24,T+2,113)
1160 CALL HCHAR(25,T+1,152)
1170 CALL HCHAR(22,15,101,3)
1180 REM  PRINT GROUND(GREEN)
1190 CALL HCHAR(23,1,128,15)
1200 CALL HCHAR(23,17,128,15)
1210 CALL HCHAR(24,1,128,14)
1220 CALL HCHAR(24,18,128,14)
1230 REM  A=HEIGHT OF PLANE
1240 RANDOMIZE
1250 A=2*INT(9*RND)+1
1260 IF A=1 THEN 1240
1270 REM  DETERMINE PLANE'S  DIRECTION
1280 RANDOMIZE
1290 B=INT(2*RND)+1
1300 ON B GOTO 1310,1350
1310 B=30
1320 CALL CHAR(96,PRS)
1330 DIR=2
1340 GOTO 1380
1350 DIR=2
1360 CALL CHAR(96,PS)
1370 B=2
1380 Y=21
1390 REM  * BEGIN LOOP *
1400 CALL KEY(0,K,S)
1410 REM  81 FIRES ROCKET.  83 AND 68
     MOVE TANK
1420 IF K=81 THEN 1450
1430 IF K=83 THEN 2640
1440 IF K=68 THEN 2770 ELSE 1550
1450 IF Y=21 THEN 1480 ELSE 1470
1460 REM  FIRE ROCKET
1470 CALL VCHAR(Y+2,T+1,32)
```

```
1480 CALL VCHAR(T,T+1,97)
1490 T=T+1
1500 T=T-2
1510 IF T=-1 THEN 1520 ELSE 1550
1520 CALL VCHAR(1,T+1,32)
1530 GOTO 1580
1540 REM  MOVE PLANE
1550 IF DIR=-2 THEN 1570
1560 IF B<5 THEN 1590
1570 CALL VCHAR(A,B-DIR,32)
1580 IF B<2 THEN 1900
1590 CALL VCHAR(A,B,96)
1600 RANDOMIZE
1610 Q=INT(15*RND)+1
1620 REM  WILL PLANE FIRE          LASER?
1630 TT=TT-2+INT(Q/2)
1640 IF B=TT THEN 1670
1650 IF Q>DIFF THEN 1670 ELSE 1820
1660 REM  PLANE WILL FIRE          LASER
1670 IF BA=5 THEN 1760 ELSE 1680
1680 IF B=16 THEN 1690 ELSE 1760
1690 BA=BA+1
1700 IF BA<>5 THEN 1720
1710 CALL HCHAR(22,15,32,3)
1720 CALL VCHAR(A+1,B,144,21-A)
1730 CALL VCHAR(A+1,B,32,21-A)
1740 CALL SOUND(500,-5,2)
1750 GOTO 1850
1760 CALL VCHAR(A+1,B,144,23-A)
1770 IF B<>T+1 THEN 1790
1780 IF A<>T+2 THEN 2440
1790 CALL VCHAR(A+1,B,32,22-A)
1800 CALL VCHAR(23,B,128,2)
1810 CALL SOUND(500,-5,2)
1820 IF B>30 THEN 1840 ELSE 1830
1830 IF B<3 THEN 1840 ELSE 1850
1840 CALL VCHAR(A,B,32)
1850 IF A<>T+2 THEN 1870
1860 IF B=T+1 THEN 2050
1870 B=B+DIR
1880 IF B>32 THEN 1900 ELSE 1890
1890 IF X=81 THEN 1420 ELSE 1400
1900 RANDOMIZE
1910 B=INT(2*RND)+1
1920 REM  DETERMINE PLANE'S    DIRECTION
1930 ON B GOTO 1940,1980
1940 B=30
1950 CALL CHAR(96,PS)
1960 DIR=-2
1970 GOTO 2000
1980 DIR=2
1990 CALL CHAR(96,PS)
2000 RANDOMIZE
2010 A=2+INT(9*RND)+1
2020 IF A=1 THEN 2000
2030 GOTO 1890
2040 REM  TANK HITS PLANE
2050 CALL SOUND(1000,-5,2)
2060 REM  A TIE-TANK WINS-     REPRINT T
     ANK
2070 CALL HCHAR(24,T+1,42)
2080 CALL HCHAR(23,T+1,152)
2090 CALL HCHAR(A,B,128)
2100 REM  DETERMINE PLANE'S    DIRECTION
2110 B=INT(2*RND)+1
2120 ON B GOTO 2130,2170
2130 B=30
2140 DIR=-2
2150 CALL CHAR(96,PRS)
2160 GOTO 2200
2170 DIR=2
2180 CALL CHAR(96,PS)
2190 REM  * SCORING *
2200 IF A>13 THEN 2210 ELSE 2230
2210 SC=SC+5
2220 GOTO 2300
2230 IF A>7 THEN 2240 ELSE 2260
2240 SC=SC+10
2250 GOTO 2300
2260 IF A<>5 THEN 2290
2270 SC=SC+15
2280 GOTO 2300
2290 SC=SC+20
2300 MS=STR$(SC)
2310 RANDOMIZE
2320 A=2*INT(9*RND)+1
2330 IF A=1 THEN 2310
2340 REM  PRINT SCORE
2350 FOR I=1 TO LEN(MS)
2360 CV=ASC(SEG$(MS,I,1))
2370 CALL HCHAR(AA,ZZ+I,CV)
2380 NEXT I
2390 GOTO 1330
2400 FOR I=1 TO 1000
2410 NEXT I
2420 GOTO 1040
2430 REM  PLANE HIT TANK
2440 CALL SOUND(500,-5,2)
2450 CALL CLEAR
2460 SX=SX+1
2470 REM  IF 2 GAMES PLAYED, START OVER
2480 IF SX<2 THEN 1040
2490 PRINT TAB(7);"  GAME OVER  "
2500 PRINT TAB(7);"YOUR SCORE IS";SC
2510 FOR I=1 TO 10
2520 PRINT
2530 NEXT I
2540 SX=0
2550 IF SC<100 THEN 2600
2560 PRINT TAB(3);"* YOU GET A FREE GAM
     E *"
2570 REM  * A FREE GAME! *
2580 SC=0
2590 GOTO 2400
2600 SC=0
2610 REM  GO BACK TO START
2620 GOTO 680
2630 REM  MOVE TANK LEFT
2640 IF T<3 THEN 2650 ELSE 2680
2650 CALL SOUND(250,110,2)
2660 T=1
2670 GOTO 2590
2680 T=T-2
2690 CALL HCHAR(24,T,112)
2700 CALL HCHAR(24,T+1,42)
2710 CALL HCHAR(24,T+2,113)
2720 CALL HCHAR(23,T+1,152)
2730 CALL HCHAR(24,T+3,128,2)
2740 CALL HCHAR(23,T+3,128)
2750 GOTO 1550
2760 REM  MOVE TANK RIGHT
2770 IF T>27 THEN 2780 ELSE 2810
2780 CALL SOUND(-250,110,2)
2790 T=29
2800 GOTO 2820
2810 T=T+2
2820 CALL HCHAR(24,T,112)
2830 CALL HCHAR(24,T+1,42)
2840 CALL HCHAR(24,T+2,113)
2850 CALL HCHAR(23,T+1,152)
2860 CALL HCHAR(24,T-2,128,2)
2870 CALL HCHAR(23,T-1,128)
2880 GOTO 1550
2890 END
```

BATTLE AT SEA

"Damn the torpedoes! Full speed ahead . . ." Get ready, all you armchair admirals out there in 99'er-land. You're about to do battle with the most crafty enemy of all—the Imperial TI Fleet. If you're old enough to remember those rainy Saturdays in the pre-TV age, you've probably spent many an hour with pencil and paper playing Battleship. In the intervening years, Battleship has been dressed up as a consumer item in many forms: First it was "cardboardized," then "plasticized," and finally "electronicized."

Well gang, as it happened, one rainy Saturday afternoon a few months ago, I had this mad urge to play Battleship . . . The expensive electronic version looked really enticing in a local toy store display, but I sure wasn't going to spring for it—especially when I had my trusty TI-99/4 personal computer waiting to carry out my every command. So program it I did. The result: Battleship has now been "99'erized" into a 16K TI BASIC version, which I call *Battle at Sea*.

Two 10 x 10 grids are displayed on the screen along with the row and column designations. The computer will ask you to enter coordinates for the placement of each of your ships on the grid at the right. Each coordinate must be entered separately; for example, first A 5 and then A 6 are entered for the destroyer. Since the ships occupy different numbers of grid squares, I've put in a counter for each ship to indicate how many remaining squares must be entered.

After all the coordinates for a ship have been entered, that ship will be displayed on the screen. Once all five ships are set up, the computer will secretly set up its own ships on the grid to the left. You won't be able to see the computer's ships, since the whole idea of the game is to try to find them.

Once the computer has set up its ships, it will ask you for the coordinates of your shot at its grid (on the left). You must enter your shot as a row letter, then a column number. Valid coordinates are from A to J and from 0 to 9. Any other entry will result in having to enter the coordinates again. Your hit or miss will be marked on the grid and displayed at the bottom of the screen as a MISS or **HIT**. The computer will then take a shot at your grid. It cannot see your ships, but it does keep track of where the hits and misses are.

After a hit, any ship that has been sunk will be displayed at the bottom of the screen. The score is also updated at this time: one point for each ship sunk. The first player to sink all five ships will win the game.

Because there are no moving objects in this game, speed was not the most important factor in the game design. The action happens to be fairly fast, but the critical factor was programming the computer to make intelligent decisions. With no limit on available memory, I might have been able to write a program with flawless logic. But here that wasn't the case—I had to stay within the confines of standard 16K TI BASIC.

I started by giving the computer a set of rules and several variables to test for a given situation. First, if a ship has been hit only once, the computer will take random shots around that hit until the direction is determined. It will then continue in that direction until the ship either sinks, misses a shot, or runs up to the edge of the grid. It will then reverse and shoot at the other end if the ship was not sunk.

And now it's you against the Imperial TI Fleet!

EXPLANATION OF THE PROGRAM
Battle At Sea

Line Nos.	
100-630	Initializaton: Set up variables, character definition, and color assignments.
640-870	Instruction page.
880-1010	Display 10 x 10 grids.
1020-1100	Control loop for setting up your ships on the 10 X 10 grid.
1110-1360	Subroutines holding data on each ship.
1370-1380	Branch to subroutine: computer sets up its ships.
1390-1530	Display message for ship coordinates to be entered.
1540-1710	Read keyboard; INPUT coordinates of ships.
1720-1950	Put the coordinates in order
1960-2050	Check that all coordinates are valid.
2060-2220	Display ship on the 10 x 10 grid.
2230-2380	Control loop holding data for computer to set up its ships.
2390-2600	Subroutine to set up computer's ships at random.
2610-2860	Set up variables for messages; subroutines for displaying those messages.
2870-2910	Keep track of which turn it is. Branch to either user's shot, or computer's shot.
2920-3170	computer takes random shot at your grid if no ships are hit.
3180-3340	Read keyboard; INPUT user's shot a computer's grid.
3350-3570	Check for valid INPUT, hit or miss.
3580-3710	Check for direction of hits on your ships.
3720-4150	Take random shot around last hit if only one hit on the ship.
4160-4450	If more than one hit on a ship takes another hit in proper direction.
4460-4620	Adjust varibales when computer gets a hit.
4630-4770	Find out how many hits on each ship; used for both computer and user.
4780-4980	Calculate score, and number of ships hit, but not sunk.
4990-5020	Display any ships that have been destroyed after every hit.
5030-5090	Display scores.
5100-5190	End of game message.
5200-5320	Re-initialize variables for next game.
5330-5340	END of game.
5350-5460	Subroutine to make sure ships are in line.

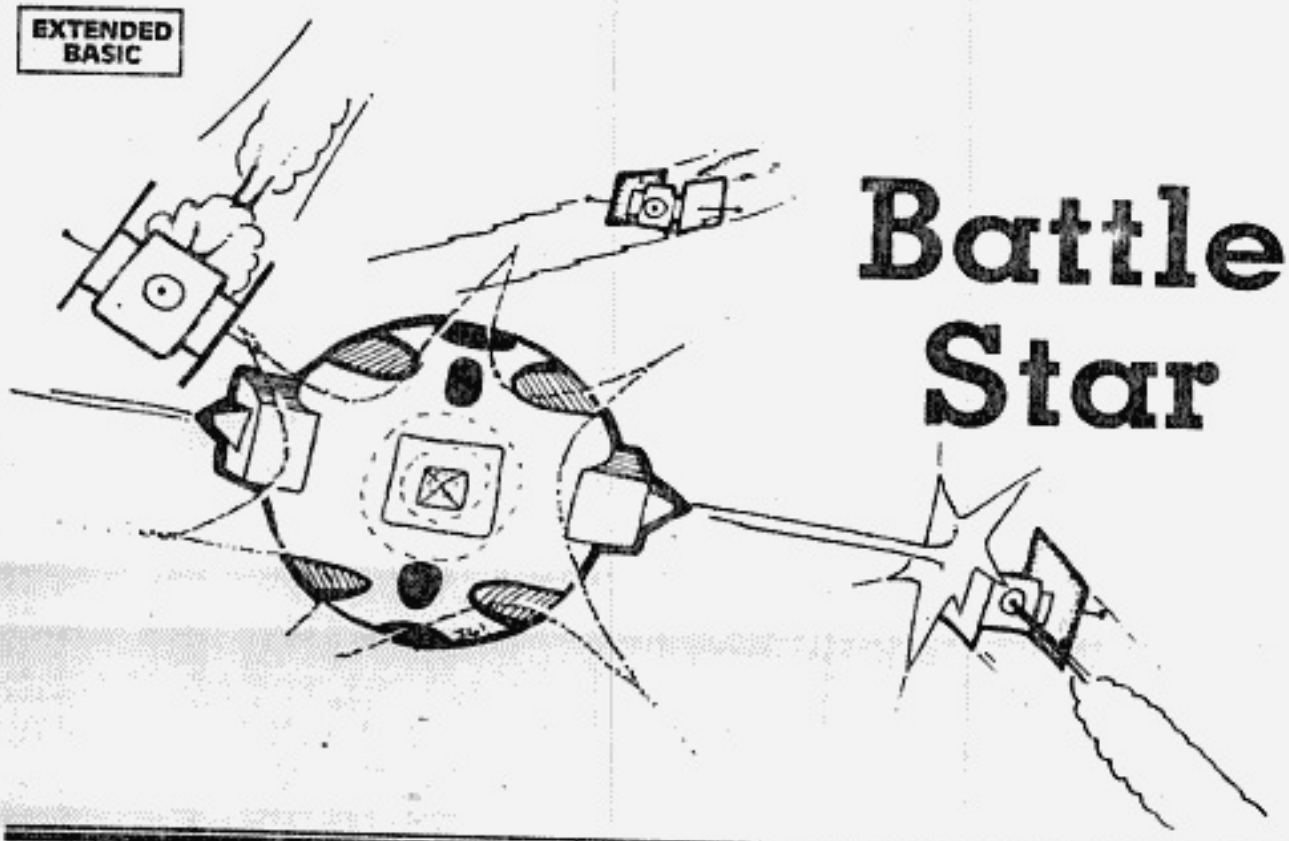
```
100 REM  *********************
110 REM  *  BATTLE AT SEA  *
120 REM  *********************
130 REM
140 REM
150 REM
160 REM
170 REM
180 RANDOMIZE
190 CALL SCREEN(12)
200 CALL CLEAR
210 PRINT TAB(7);"BATTLE AT SEA"
220 PRINT
230 PRINT
240 PRINT ::::::::::::::::::::
250 OPTION BASE 1
260 DIM P(10,10),O(10,10),SH(5,5,2)
270 CALL COLOR(14,7,1)
280 CALL COLOR(15,11,1)
290 CALL CHAR(96,"FFFFFFFFF7F3F1F0")
300 CALL CHAR(97,"00000000FFFFFFFF")
310 CALL CHAR(98,"3C7EFFFFFFFFFFFF")
320 CALL CHAR(99,"000000FFFEFCF880")
330 CALL CHAR(100,"103070707070707070")
340 CALL CHAR(101,"707878787878787878")
350 CALL CHAR(102,"787C7E7E7F7F7C7800")
360 CALL CHAR(103,"707070707070701010")
370 CALL CHAR(104,"00000040307F7F3F00")
380 CALL CHAR(105,"8C4C3CFEFFFFFF00")
390 CALL CHAR(106,"0102030FFFFFFF00")
400 CALL CHAR(107,"0002040F8FFFFEFE")
410 CALL CHAR(108,"103070727478787878")
420 CALL CHAR(109,"7C7C787174707C7C")
430 CALL CHAR(110,"7F7F787C7C7C7A78")
440 CALL CHAR(111,"707878787C7C727110")
450 CALL CHAR(112,"00108867FF7F3F00")
460 CALL CHAR(113,"09C5C3F3FFFFFF00")
470 CALL CHAR(114,"0002047F8FFFFEFE")
480 CALL CHAR(115,"103070727478787A7C")
490 CALL CHAR(116,"787A7C7C7F7F787C00")
500 CALL CHAR(117,"7C7C7A7078787C7C1A")
510 CALL CHAR(118,"00000001FFFFFF3F00")
520 CALL CHAR(119,"867EFEFFFFFFFFFE00")
530 CALL CHAR(120,"0000000180C7FFFE00")
540 CALL CHAR(121,"103070787878787878")
550 CALL CHAR(122,"787C7C7E7E7E7F7F00")
560 CALL CHAR(123,"7C7C787878703020")
570 CALL CHAR(124,"03030F1FFFFF3F00")
580 CALL CHAR(125,"0006060F8FFFFFFE00")
590 CALL CHAR(126,"103070707878787E7F")
600 CALL CHAR(127,"7C7878787070707010")
610 CALL CHAR(128,"FF8181818181818181FF")
620 CALL CHAR(136,"815E2C366A3C2442")
630 CALL CHAR(144,"8166673C3C7E6681")
640 CALL SOUND(-3000,220,30,554,20,194
    7,20,-8,30)
650 PRINT TAB(7);"BATTLE AT SEA"
660 PRINT :"YOU MUST DESTROY THE ENEMY
    "
670 PRINT "SHIPS BEFORE THE COMPUTER"
680 PRINT "DESTROYS YOUR SHIPS."
690 PRINT :"TO SET UP YOUR SHIPS YOU"
700 PRINT "MUST ENTER COORDINATES ON"
710 PRINT "THE 10 X 10 GRID ON THE
    RIGHT."
720 PRINT "ENTER THE ROW, THEN THE
    COLUMN."
730 PRINT "EXAMPLE: AS"
740 PRINT "AFTER YOUR SHIPS ARE SET UP
    ."
750 PRINT "YOU WILL TAKE A SHOT AT THE
    "
760 PRINT "ENEMY SHIPS BY ENTERING ONE
    "
770 PRINT "PAIR OF COORDINATES ON THE"
780 PRINT "ENEMY GRID."
790 PRINT "THE COMPUTER WILL THEN"
800 PRINT "TAKE A SHOT AT YOUR SHIPS."
810 PRINT :"THE COMPUTER CANNOT SEE"
820 PRINT "YOUR SHIPS. YOU CANNOT SEE"
830 PRINT "THE COMPUTER'S SHIPS."
840 PRINT "ENTER ANY KEY TO BEGIN."
850 CALL SOUND(11,-2,5)
860 CALL KEY(0,K,S)
870 IF S=0 THEN 860
880 CALL SCREEN(6)
890 CALL CLEAR
900 PRINT "   COMPUTER          YOU"
910 PRINT ::::::::::::::::::::::::::
920 FOR X=5 TO 14
930 CALL VCHAR(X,5,X+60)
940 CALL HCHAR(X,6,128,10)
950 CALL HCHAR(X,18,128,10)
960 CALL VCHAR(X,17,X+60)
970 NEXT X
980 FOR X=6 TO 15
990 CALL VCHAR(15,X+12,X+42)
1000 CALL VCHAR(15,X,X+42)
1010 NEXT X
1020 S1$="CARRIER"
1030 S2$="BATTLESHIP"
1040 S3$="CRUISER"
```

```
1050 SUS$="SUBMARINE"
1060 SS$="DESTROYER"
1070 FOR S=1 TO 5
1080 ON S GOSUB 1110,1160,1210,1260,131
     0
1090 GOSUB 1390
1100 GOTO 1360
1110 PRS=SS$
1120 LE=5
1130 S=1
1140 OS=0
1150 RETURN
1160 PRS=S2$
1170 LE=4
1180 S=2
1190 OS=8
1200 RETURN
1210 PRS=S3$
1220 LE=3
1230 S=3
1240 OS=16
1250 RETURN
1260 PRS=S4$
1270 LE=5
1280 S=4
1290 OS=22
1300 RETURN
1310 PRS=S5$
1320 LE=2
1330 S=5
1340 OS=28
1350 RETURN
1360 NEXT S
1370 CALL HCHAR(22,1,52,64)
1380 GOTO 2240
1390 L=LEN(PRS)
1400 SUS$="ENTER ROW,COL. FOR "&STR$(LE
     N)&" SPACES."
1410 FOR X=1 TO LEN(SUS$)
1420 SU1$=SEG$(SUS$,X,1)
1430 CALL VCHAR(22,X+2,ASC(SU1$))
1440 NEXT X
1450 PRS=PRS&" SPACE"
1460 CALL HCHAR(23,2,32,30)
1470 FOR X=1 TO LEN(PRS)
1480 SU1$=SEG$(PRS,X,1)
1490 CALL VCHAR(23,X+2,ASC(SU1$))
1500 NEXT X
1510 FOR X=1 TO LE
1520 CALL HCHAR(23,20,35)
1530 CALL VCHAR(23,21,LE-X+48)
1540 CALL KEY(0,K1,ST)
1550 IF ST=0 THEN 1540
1560 IF K1<65 THEN 1590
1570 IF K1>74 THEN 1590
1580 GOTO 1610
1590 CALL SOUND(100,-2,2)
1600 GOTO 1540
1610 CALL VCHAR(23,23,K1)
1620 CALL KEY(0,KE,ST)
1630 IF ST=1 THEN 1620
1640 CALL KEY(0,K2,ST)
1650 IF ST=0 THEN 1640
1660 IF K2<48 THEN 1690
1670 IF K2>57 THEN 1690
1680 GOTO 1710
1690 CALL SOUND(100,-2,2)
1700 GOTO 1640
1710 CALL VCHAR(23,24,K2)
1720 SH(S,X,1)=K1-64
1730 SH(S,X,2)=K2-47
1740 IF P(K1-64,K2-47)>0 THEN 1590
1750 P(K1-64,K2-47)=S
1760 NEXT X
1770 GOSUB 5370
1780 IF SH(S,1,1)=SH(S,2,1)THEN 1810
1790 X2=1
1800 GOTO 1820
1810 X2=2
1820 FOR X1=1 TO LE
1830 F=0
1840 FOR X1=1 TO LE-X1
1850 IF SH(S,X1,X2)=0 THEN 1910
1860 IF SH(S,X1,X2)<SH(S,X1+1,X2)THEN 1
     910
1870 SW=SH(S,X1,X2)
1880 SH(S,X1,X2)=SH(S,X1+1,X2)
1890 SH(S,X1+1,X2)=SW
1900 F=1
1910 NEXT X1
1920 IF F=0 THEN 1940
1930 NEXT X1
1940 FOR X=1 TO LE-1
1950 IF SH(S,X,1)<>SH(S,X+1,1)-1 THEN 1
     980
1960 NEXT X
1970 GOTO 2070
1980 FOR X=1 TO LE-1
1990 IF SH(S,X,2)<>SH(S,X+1,2)-1 THEN 2
     020
2000 NEXT X
2010 GOTO 2070
2020 CALL SOUND(100,-2,2)
2030 FOR X=1 TO LE
2040 P(SH(S,X,1),SH(S,X,2))=0
2050 NEXT X
2060 GOTO 1460
2070 X=S
2080 FOR X1=1 TO 5
2090 IF SH(X,X1,1)=0 THEN 2190
2100 IF SH(X,1,1)=SH(X,2,1)THEN 2130
2110 OSA=1
2120 GOTO 2140
2130 OSA=0
2140 P(SH(X,X1,1),SH(X,X1,2))=X
2150 IF X>1 THEN 2180
2160 CALL VCHAR(SH(X,X1,1)+6,SH(X,X1,2)
     +17.95+X1+OS+((LE-1)*OSA))
2170 GOTO 2190
2180 CALL HCHAR(SH(X,X1,1)+6,SH(X,X1,2)
     +17.95+X1+OS+((LE*OSA))
2190 NEXT X1
2200 IF X>1 THEN 2230
2210 CALL HCHAR(SH(1,4,1)+6,SH(1,4,2)+1
     7.97+((LE-1)*OSA))
2220 CALL HCHAR(SH(1,5,1)+6,SH(1,5,2)+1
     7.99+((LE-1)*OSA))
2230 RETURN
2240 LE=4
2250 S=1
2260 GOSUB 2400
2270 LE=5
2280 S=2
2290 GOSUB 2400
2300 LE=2
2310 S=3
2320 GOSUB 2400
2330 LE=2
2340 S=4
2350 GOSUB 2400
2360 LE=1
2370 S=5
2380 GOSUB 2400
2390 GOTO 2630
2400 RANDOMIZE
2410 X2=INT(RND*2)+1
2420 IF X2=2 THEN 2460
2430 X=INT(RND*(10-LE))+1
2440 X1=INT(RND*10)+1
2450 GOTO 2480
2460 X=INT(RND*10)+1
2470 X1=INT(RND*(10-LE))+1
2480 ON X2 GOTO 2490,2560
2490 FOR Y=X TO X+LE
2500 IF O(Y,X1)>0 THEN 2400
2510 NEXT Y
2520 FOR Y=X TO X+LE
2530 O(Y,X1)=S
```

```
2540 NEXT T
2550 RETURN
2560 FOR T=X1 TO X1+LE
2570 IF O(X,T)>0 THEN 2400
2580 NEXT T
2590 FOR T=X1 TO X1+LE
2600 O(X,T)=S
2610 NEXT T
2620 RETURN
2630 M1$="MY SHOT"
2640 M2$="YOUR SHOT"
2650 M3$="SCORE"
2660 M4$="COMPUTER"
2670 M5$="USER"
2680 M6$="YOU MISSED"
2690 M7$="I MISSED"
2700 M8$="**HIT**"
2710 GOTO 2800
2720 FOR V=1 TO 7
2730 CALL HCHAR(18,V+4,ASC(SEG$(M1$,V,1
))))
2740 NEXT V
2750 RETURN
2760 FOR V=1 TO 9
2770 CALL HCHAR(21,V+4,ASC(SEG$(M2$,V,1
))))
2780 NEXT V
2790 RETURN
2800 FOR X=1 TO 5
2810 CALL HCHAR(18,X+22,ASC(SEG$(M3$,X,
1))))
2820 NEXT X
2830 FOR X=1 TO 8
2840 CALL HCHAR(19,X+15,ASC(SEG$(M4$,X,
1))))
2850 NEXT X
2860 FOR X=1 TO 4
2870 CALL HCHAR(19,X+26,ASC(SEG$(M5$,X,
1))))
2880 NEXT X
2890 T=1
2900 IF T=0 THEN 2930
2910 T=0
2920 GOTO 3200
2930 T=1
2940 CALL HCHAR(21,3,32,12)
2950 CALL HCHAR(22,3,32,7)
2960 GOSUB 2720
2970 IF W>0 THEN 3650
2980 RANDOMIZE
2990 X=INT(10*RND)+1
3000 X1=INT(10*RND)+1
3010 H=X
3020 H1=X1
3030 IF P(X,X1)=7 THEN 2980
3040 IF P(X,X1)=6 THEN 2980
3050 CALL HCHAR(19,6,H+64)
3060 CALL HCHAR(19,7,H1+47)
3070 IF P(X,X1)>0 THEN 4480
3080 GOSUB 3120
3090 GOTO 2980
3100 P(X+10,X1)=7
3110 CALL HCHAR(23,1,32,32)
3120 P(X,X1)=6
3130 CALL SOUND(200,-6,2)
3140 CALL HCHAR(23,1,32,32)
3150 CALL VCHAR(X+4,X1+17,144)
3160 FOR T=1 TO 8
3170 CALL VCHAR(23,12+T,ASC(SEG$(M7$,T,
1))))
3180 NEXT T
3190 RETURN
3200 CALL HCHAR(18,3,32,12)
3210 CALL HCHAR(19,3,32,7)
3220 GOSUB 2760
3230 CALL KEY(0,K1,ST)
3240 IF ST=0 THEN 3230
3250 IF K1<65 THEN 3230
3260 IF K1>74 THEN 3230
3270 CALL VCHAR(22,6,K1)
3280 CALL KEY(0,KE,ST)
3290 IF ST=1 THEN 3280
3300 CALL KEY(0,K2,ST)
3310 IF ST=0 THEN 3300
3320 IF K2<48 THEN 3300
3330 IF K2>57 THEN 3300
3340 CALL VCHAR(22,7,K2)
3350 K3=K1-64
3360 K4=K2-47
3370 IF O(K3,K4)<6 THEN 3410
3380 CALL SOUND(50,110,2)
3390 CALL HCHAR(22,6,32,7)
3400 GOTO 3200
3410 IF O(K3,K4)=0 THEN 3520
3420 CALL SOUND(200,220,2,330,2,440,2,-
6,2)
3430 CALL SOUND(400,110,2,220,2,330,2,-
8,2)
3440 CALL VCHAR(K3+4,K4+5,136)
3450 SF=O(K3,K4)
3460 O(K3,K4)=7
3470 CALL HCHAR(23,1,32,32)
3480 FOR X2=1 TO 7
3490 CALL HCHAR(23,13+X2,ASC(SEG$(M8$,X
2,1)))
3500 NEXT X2
3510 GOTO 4620
3520 CALL SOUND(200,-6,2)
3530 CALL HCHAR(23,1,32,32)
3540 O(K3,K4)=6
3550 FOR X2=1 TO 10
3560 CALL VCHAR(23,13+X2,ASC(SEG$(M6$,X
2,1)))
3570 NEXT X2
3580 CALL VCHAR(K3+4,K4+5,144)
3590 GOTO 2990
3600 CH=1
3610 GOTO 4670
3620 CH=0
3630 ON SF GOSUB 1110,1160,1210,1260,13
10
3640 IF DS(ST)=LE-1 THEN 3800
3650 IF H=10 THEN 3690
3660 IF P(H+1,H1)<>7 THEN 3680
3670 IF W>1 THEN 4280 ELSE 4080
3680 IF H=1 THEN 3740
3690 IF P(H-1,H1)<>7 THEN 3740
3700 IF W>1 THEN 4280 ELSE 4080
3710 W2=W
3720 W=W1
3730 GOTO 3580
3740 IF H1=10 THEN 3780
3750 IF P(H,H1+1)<>7 THEN 3770
3760 IF W>1 THEN 4080 ELSE 4280
3770 IF H1=1 THEN 3800
3780 IF P(H,H1-1)<>7 THEN 3800
3790 IF W>1 THEN 4080 ELSE 4280
3800 L1=INT(RND*2)+1
3810 ON L1 GOTO 3820,3900
3820 X2=INT(RND*2)+1
3830 ON X2 GOTO 3840,3870
3840 X2=1
3850 X5=0
3860 GOTO 3970
3870 X2=-1
3880 X5=0
3890 GOTO 3970
3900 X5=INT(RND*2)+1
3910 ON X5 GOTO 3920,3950
3920 X5=1
3930 X2=0
3940 GOTO 3970
3950 X5=-1
3960 X2=0
3970 IF H+X2>10 THEN 3800
3980 IF H+X2<1 THEN 3800
3990 IF H1+X3>10 THEN 3800
4000 IF H1+X3<1 THEN 3800
```

```
4010 IF P(H+X2,H1+X5)=6 THEN 3800
4020 IF P(H+X2,H1+X5)=7 THEN 3800
4030 H=H+X2
4040 X1=H1+X5
4050 IF P(H,X1)>0 THEN 4480
4060 GOSUB 3120
4070 GOTO 2900
4080 IF H=10 THEN 4180
4090 H=H+1
4100 IF P(H,H1)=7 THEN 4080
4110 IF P(H,H1)=6 THEN 4100
4120 X=H
4130 X1=H1
4140 IF P(X,X1)>0 THEN 4480
4150 GOSUB 3120
4160 H=H-1
4170 GOTO 2900
4180 IF H=1 THEN 4090
4190 H=H-1
4200 IF P(H,H1)=7 THEN 4180
4210 IF P(H,H1)=6 THEN 4080
4220 X=H
4230 X1=H1
4240 IF P(X,X1)>0 THEN 4480
4250 GOSUB 3120
4260 H=H+1
4270 GOTO 2900
4280 IF H1=10 THEN 4380
4290 H1=H1+1
4300 IF P(H,H1)=7 THEN 4280
4310 IF P(H,H1)=6 THEN 4380
4320 X=H
4330 X1=H1
4340 IF P(X,X1)>0 THEN 4480
4350 GOSUB 3120
4360 H1=H1-1
4370 GOTO 2900
4380 IF H1=1 THEN 4280
4390 H1=H1-1
4400 IF P(H,H1)=7 THEN 4380
4410 IF P(H,H1)=6 THEN 4280
4420 X=H
4430 X1=H1
4440 IF P(X,X1)>0 THEN 4480
4450 GOSUB 3120
4460 H1=H1+1
4470 GOTO 2900
4480 CALL VCHAR(4+X,17+X1,136)
4490 CALL HCHAR(23,1,32,32)
4500 GOSUB 2720
4510 FOR Z=1 TO LEN(MB$)
4520 CALL HCHAR(23,14+Z,ASC(SEGS(MB$,Z,
1)))
4530 NEXT Z
4540 CALL SOUND(200,220,2,330,2,440,2,-
8,2)
4550 CALL SOUND(300,110,0,220,0,330,0,-
8,0)
4560 SF=P(X,X1)
4570 CALL VCHAR(19,6,X+64)
4580 CALL VCHAR(19,7,X1+47)
4590 P(X,X1)=7
4600 H=X
4610 H1=X1
4620 FOR X2=1 TO 5
4630 DS(X2)=0
4640 NEXT X2
4650 FOR X2=1 TO 10
4660 FOR X3=1 TO 10
4670 IF CH=1 THEN 4690
4680 IF T=0 THEN 4740
4690 IF P(X2,X3)=0 THEN 4780
4700 IF P(X2,X3)=6 THEN 4780
4710 IF P(X2,X3)=7 THEN 4780
4720 DS(P(X2,X3))=DS(P(X2,X3))+1
4730 GOTO 4780
4740 IF O(X2,X3)=0 THEN 4780
4750 IF O(X2,X3)=6 THEN 4780
4760 IF O(X2,X3)=7 THEN 4780
4770 DS(O(X2,X3))=DS(O(X2,X3))+1
4780 NEXT X3
4790 NEXT X2
4800 IF CH=1 THEN 3620
4810 W=0
4820 SCORE=0
4830 FOR Z4=1 TO 5
4840 ON Z4 GOSUB 1110,1160,1210,1260,13
10
4850 IF DS(Z4)=LE THEN 4940
4860 IF DS(Z4)=0 THEN 4890
4870 W=W+1
4880 GOTO 4940
4890 SCORE=SCORE+1
4900 IF T=0 THEN 4930
4910 GOSUB 5010
4920 GOTO 4940
4930 GOSUB 5000
4940 NEXT Z4
4950 IF T=0 THEN 4980
4960 W1=W
4970 GOTO 2900
4980 W=W1
4990 GOTO 2900
5000 SCP=SCORE
5010 CALL HCHAR(23,1,32,32)
5020 FOR X3=1 TO LEN(PR$)+10
5030 CALL HCHAR(23,X3+6,ASC(SEGS(PR$,
DESTROYED",X3,1)))
5040 NEXT X3
5050 IF T=0 THEN 5090
5060 CALL VCHAR(20,20,SCORE+48)
5070 IF SCORE=5 THEN 5120
5080 RETURN
5090 CALL HCHAR(20,27,SCORE+48)
5100 IF SCORE=5 THEN 5140
5110 RETURN
5120 PRINT "THE COMPUTER WINS AGAIN"
5130 GOTO 5150
5140 PRINT "YOU JUST GOT LUCKY THIS TIM
E."
5150 PRINT "IF YOU WISH TO PLAY AGAIN"
5160 PRINT "ENTER "Y", IF NOT ENTER "
N"."
5170 INPUT NG$
5180 IF NG$="N" THEN 5550
5190 IF NG$="Y" THEN 5220
5200 CALL SOUND(200,110,0)
5210 GOTO 5150
5220 FOR L=1 TO 10
5230 FOR L1=1 TO 10
5240 P(L,L1)=0
5250 O(L,L1)=0
5260 NEXT L1
5270 NEXT L
5280 FOR L=1 TO 5
5290 FOR L1=1 TO 5
5300 SH(L,L1,1)=0
5310 SH(L,L1,2)=0
5320 NEXT L1
5330 NEXT L
5340 GOTO 800
5350 CALL CLEAR
5360 STOP
5370 NNN=0
5380 AAA=0
5390 FOR X=1 TO LE-1
5400 IF NNN=1 THEN 5430
5410 IF SH(S,X,1)=SH(S,X+1,1)THEN 5460
5420 IF AAA=1 THEN 2020
5430 IF SH(S,X,2)<>SH(S,X+1,2)THEN 2020
5440 NNN=1
5450 GOTO 5470
5460 AAA=1
5470 NEXT X
5480 RETURN
5490 END
```

EXTENDED BASIC



Battle Star

You are the chief security officer in charge of defending the Earth's newest Battle Star from alien attack. At first, the aliens are few—trying only to probe your weak points. Later, they attack in force from all four directions. It's their somewhat ancient nuclear missiles against your laser battery. One hit by a missile, however, and the entire Battle Star is obliterated. The speed at which you can react and move your fingers is the only thing that stands between victory and total destruction . . .

To fire a laser in any one of four directions, press any of the arrow keys. These are the only keys used. You may not move your Battle Star because of your geosynchronous orbit and large size. The entire game is an hand-eye coordination exercise. At one point in the game, the aliens become so fast you may not be able to move fast enough to prevent annihilation. There is, however, an "automatic speed check" put into the program; if you can reach this level and maintain it, the endurance of your fingers will be your only limiting factor. If you wish to make the game even more difficult, you could adjust the limiting speed of the missiles. This is done in lines 730, 760, 790, and 820. The X and Y velocity in the sprite being defined (whichever X or Y is not zero) can be adjusted. For example, in line 730 the X velocity formula is 11 – (L/10). This will allow no speed greater than 10. Change this to 15 – (L/10) and the maximum speed will be 14, with the initial speed being 5. If one line is changed, related lines must all be changed.

The Program

The program is very short and simple—requiring only 3K memory and Extended BASIC. There is plenty of room for a good programmer to experiment and try adding to or improving the features. The action is simple, but can become fairly rapid—thus making the game very challenging. A Battle Star is positioned at the center of the screen, and made up of 9 sprites (3x3). I did this for dramatic reasons: When the Battle Star is hit, each section of it blows up and flies in a different direction. An alien ship will appear to the left, right, above, and below the Battle Star. At first, only the ship will be displayed; later, the ship and a nuclear missile will appear. For every missile knocked out of action, your score will increase by 20 points. For every alien ship destroyed, you will receive 50 points. The trickiest part of the program was to make the laser rays coming from the Battle Star stop after encountering a missile. Since the missile is a sprite, its location is checked using the CALL POSITION statement. Then, calculating the distance from the Battle Star's cannon and dividing by 9 gives me the distance (in number of characters) of the missile. I then use a CALL HCHAR, or CALL VCHAR (first with the ray bolt) and a CHR$(32) which represents a space. Finally, I delete the given sprite. The result is a fast laser bolt and increased program speed.

One problem I encountered when the missiles were traveling at high speed was that they sometimes passed through the base without a hit being detected. This problem was alleviated by checking POSITION instead of COINC, so that if the position was past the edge of the Battle Star, the missile would blow up.

EXPLANATION OF THE PROGRAM

Battle Star

Line Nos.	
170-290	Initializes colors and characters.
300	Initializes variables.
310	Jumps to subroutine to create Battle Star.
320-340	Main program loop.
350-380	Sets up sprites to create the Battle Star.
390-440	Reads keyboard; branches to fire laser cannon.
450-490	Fires laser up.
500-540	Fires laser left.
550-590	Fires laser down.
600-640	Fires laser right.
650-690	Checks position of missiles, and branches off if Battle Star hit.
700	Checks the chance of another ship appearing.
710	Decides which ship will appear, and branches to subroutine.
720-740	Places top ship on screen with missile if game progressed.
750-770	Places left ship on screen with missile if game progressed.
780-800	Places bottom ship on screen with missile if game progressed.
810-830	Places right ship on screen with missile if game progressed.
840-870	Battle Star is hit and destroyed.
880-910	Displays score. Play again? Accepts answer.
920-940	Re-initializes variables.
950	End.

```
100 REM *-*-*-*-*-*-*-*-*-*-*-*-*
110 REM *                       *
120 REM *  BATTLE STAR           *
130 REM *                       *
140 REM *-*-*-*-*-*-*-*-*-*-*-*-*
150 REM
160 REM
170 RANDOMIZE
180 DIR=-1 :: CALL CLEAR
190 CALL COLOR(9,7,1):: CALL COLOR(10,
6,1):: CALL SCREEN(2)
200 CALL CHAR(96,"0000000000707070"):: 
  CALL CHAR(97,"183C7EFFDB991818")
210 CALL CHAR(98,"0000000000E0E0E0"):: 
  CALL CHAR(99,"070E1CFFFF1C0E07"):: 
220 CALL CHAR(100,"18423C9999C34218"):: 
  :: CALL CHAR(101,"E0703CFFFF3C70E0"
  ):: CALL CHAR(102,"0707070707")
230 CALL CHAR(107,"1046282440A92304")
240 CALL CHAR(105,"99DBFF7E3C181818"):: 
  :: CALL CHAR(108,"E0E0E0")
250 CALL CHAR(112,"307878C477C78300"):: 
  CALL CHAR(113,"1010103C6EEEEE7C"):: 
260 CALL CHAR(114,"0CC1E3EE23E1E00C"):: 
  CALL CHAR(115,"007CEEEE6C381010")
270 CALL CHAR(116,"1010387E381010"):: 
  CALL CHAR(117,"0000183CFF7E2442"):: 
280 CALL CHAR(105,"18181818181818"):: 
  :: CALL CHAR(106,"FFFFFF")
290 FOR COL=1 TO 12 :: CALL COLOR(COL,
16,1):: NEXT COL
300 L=100 :: S=5 :: SC=0 :: SA1,SB1,SA
2,SB2,SA3,SB3,SA4,SB4=0 :: T=0
310 GOSUB 350
320 GOSUB 390 :: GOSUB 650
330 L=L-S :: IF L<1 THEN L=1
340 DISPLAY AT(24,3):SC :: GOTO 320
350 CALL SPRITE(#10,96,16,81,113,0,0,#
11,97,16,81,121,0,0,#12,98,16,81,1
29,0,0)
360 CALL SPRITE(#13,99,16,89,113,0,0,#
14,104,7,89,121,0,0,#15,101,16,89,
129,0,0)
370 CALL SPRITE(#16,102,16,97,113,0,0,
#17,103,16,97,121,0,0,#18,100,16,9
7,129,0,0)
380 RETURN
390 CALL KEY(0,K,S):: IF S=0 THEN RETU
RN
400 IF K=69 THEN 450
410 IF K=83 THEN 500
420 IF K=88 THEN 550
430 IF K=68 THEN 600
440 RETURN
450 IF SA1=0 AND SB1=0 THEN CALL VCHAR
(11,16,105,10):: CALL SOUND(10,800,
0):: CALL VCHAR(1,16,32,10):: SC=S
C-10 :: RETURN
460 IF SB1=0 THEN CALL VCHAR(2,16,105,
9):: CALL SOUND(500,110,2,-5,2):: 
CALL VCHAR(2,16,32,9):: SC=SC+50 :
: SA1=0 :: RETURN
470 CALL POSITION(#1,P1,P2):: IF P1>76
  THEN 840
480 P1=INT(P1/8)+1 :: CALL VCHAR(P1,16
,105,10-P1):: CALL SOUND(200,110,1
0,-5,8):: CALL VCHAR(P1,16,32,10-P
1)
490 CALL DELSPRITE(#1):: SC=SC+20 :: S
A1=0 :: RETURN
500 IF SA2=0 AND SB2=0 THEN CALL HCHAR
(12,1,106,14):: CALL SOUND(10,800,
0):: CALL HCHAR(12,1,32,14):: SC=S
C-10 :: RETURN
510 IF SB2=0 THEN CALL HCHAR(12,3,106,
12):: CALL SOUND(500,110,2,-5,2):: 
  CALL HCHAR(12,3,32,12):: SC=SC+50
  :: SA2=0 :: RETURN
520 CALL POSITION(#2,P1,P2):: IF P2>86
  THEN 840
530 P2=INT(P2/8)+1 :: CALL HCHAR(12,P2
,106,15-P2):: CALL SOUND(200,110,1
0,-5,8):: CALL HCHAR(12,P2,32,15-P
2)
540 CALL DELSPRITE(#2):: SC=SC+20 :: S
B2=0 :: RETURN
550 IF SA3=0 AND SB3=0 THEN CALL VCHAR
(14,16,105,10):: CALL SOUND(10,800,
0):: CALL VCHAR(14,16,32,10):: SC
=SC-10 :: RETURN
560 IF SB3=0 THEN CALL VCHAR(14,16,105
,10):: CALL SOUND(500,110,2,-5,2):: 
  CALL VCHAR(14,16,32,10):: SC=SC+
50 :: SA3=0 :: RETURN
570 CALL POSITION(#3,P1,P2):: IF P1<11
0 AND P1>0 THEN 840
580 P1=INT(P1/8)+1 :: CALL VCHAR(14,16
,105,P1-14):: CALL SOUND(200,110,1
0,-5,8):: CALL VCHAR(14,16,32,P1-1
4)
590 CALL DELSPRITE(#3):: SC=SC+20 :: S
B3=0 :: RETURN
600 IF SA4=0 AND SB4=0 THEN CALL HCHAR
(12,18,106,14):: CALL SOUND(10,800,
0):: CALL HCHAR(12,18,32,14):: SC
=SC-10 :: RETURN
610 IF SB4=0 THEN CALL HCHAR(12,18,106
,13):: CALL SOUND(500,110,2,-5,2):: 
  CALL HCHAR(12,18,32,13):: SC=SC+
50 :: SA4=0 :: RETURN
620 CALL POSITION(#4,P1,P2):: IF P2<14
2 AND P2>0 THEN 840
630 P2=INT(P2/8):: CALL HCHAR(12,18,10
6,P2-15):: CALL SOUND(200,110,10,-
5,8):: CALL HCHAR(12,18,32,P2-15)
```

```
640 CALL DELSPRITE(#4)::  SC=SC+20  ::  S
    S4=0  ::  RETURN
650 IF S$1=0 THEN P1,P2=0  ::  GOTO  660
    ELSE CALL POSITION(#1,P1,P2)
660 IF S$2=0 THEN P3,P4=0  ::  GOTO  670
    ELSE CALL POSITION(#2,P3,P4)
670 IF S$3=0 THEN P5,P6=0  ::  GOTO  680
    ELSE CALL POSITION(#3,P5,P6)
680 IF S$4=0 THEN P7,P8=0  ::  GOTO  690
    ELSE CALL POSITION(#4,P7,P8)
690 IF P1>76 OR P4>80 OR(P5<110 AND P5
    >90)OR(P8<142 AND P8>0)THEN 860
700 RN=INT(RND*L)  ::  IF RN>10 THEN RETU
    RN
710 RN=INT(RND*4)+1  ::  ON RN GOTO 730,
    760,790,820
720 IF SA1=1 AND SB1=1 THEN RETURN
730 CALL HCHAR(2,15,115)  ::  SA1=1  ::  IF
     L<80 AND S$1=0 THEN CALL SPRITE(#
    1,116,7,17,120,11-(L/10),0)  ::  SB1=
    1
740 RETURN
750 IF SA2=1 AND SB2=1 THEN RETURN
760 CALL HCHAR(12,3,112)  ::  SA2=1  ::  IF
     L<80 AND S$2=0 THEN CALL SPRITE(#
    2,116,7,88,17,0,11-(L/10))  ::  SB2=1
770 RETURN
780 IF SA3=1 AND SB3=1 THEN RETURN
790 CALL HCHAR(23,16,115)  ::  SA3=1  ::  I
    F L<80 AND S$3=0 THEN CALL SPRITE(
    #3,116,7,175,120,-11+(L/10),0)  ::  S
    B3=1
```

```
800 RETURN
810 IF SA4=1 AND SB4=1 THEN RETURN
820 CALL HCHAR(12,30,114)  ::  SA4=1  ::  I
    F L<80 AND SB4=0 THEN CALL SPRITE(
    #4,116,7,88,216,0,-11+(L/10))  ::  SB
    4=1
830 RETURN
840 CALL DELSPRITE(#1,#2,#3,#4)  ::  CALL
     SOUND(2000,110,2,220,2,1000,30,-4
    ,2)
850 FOR BU=10 TO 15  ::  CALL MOTION(#B
    U,INT(RND*40)-20,INT(RND*40)-20)::
    CALL PATTERN(#BU,107)  ::  NEXT BU
860 CALL SOUND(1000,110,2,220,2,110,2,
    -5,2)  ::  CALL SOUND(1,40000,30)
870 CALL DELSPRITE(ALL)  ::  CALL CLEAR
880 DISPLAY AT(12,7):"YOUR SCORE IS":T
    AB(10);SC
890 CALL DELSPRITE(ALL)
900 DISPLAY AT(22,1):"DO YOU WISH TO P
    LAY AGAIN? (Y/N)."
910 ACCEPT AT(23,8)VALIDATE("YN"):ANS$
    ::  IF ANS$="N" THEN 950
920 CALL CLEAR  ::  GOSUB 550  ::  SC=0  ::
     L=100
930 CALL SB1,SB2,SB3,SB4,P1,P2,P5,P6,P
    7,P8=0
940 RETURN
950 END
```

TI BASIC

The HARRIED HOUSEWIFE

This matching game is dedicated to tired housewives everywhere who face the daily battle of keeping their houses clean amidst the unrelenting attacks from their kids, husbands, dogs, cats, visiting relatives, unexpected friends, and even home computers—those new family additions that seem to be forever spawning dust, out-of-place furniture, and loose papers.

Harried Housewife uses the color graphics of TI BASIC to depict eight household chores: dusting, sewing, washing clothes, doing dishes, cooking, vacuuming, shopping, and ironing. It is a matching game that even young children will enjoy playing. The rules are simple: An array of 16 squares is displayed on the screen. Each square represents one of the eight chores, and there are two of each chore somewhere in the array. The object of the game is to find each pair. You do this by choosing two squares at a time and entering the corresponding two letters. As a letter is entered, the chore for that square is shown. If a match is made, the chore is considered finished and is listed on the right side of the screen. If a match is not made, the two selections are covered, and two more letters may be chosen.

When all eight pairs are matched, the housework is complete; you have a clean house and the game is over. But you mustn't take too long, because when the kids come home (determined by the counter in line 1420) everything gets scrambled and the harried housewife must start over. . . And as all harried housewives undoubtedly know: it's not easy to get a completely clean house. Often the goal is to become somewhat more attainable—just seeing how it can be accomplished before the kids come home.

If you get too harried and want to quit, press S for stop. The arrangement of the current array will be displayed. After you have examined it, SHIFT C (BREAK) to end the program. If you really feel you must win more often — that is, end up with everything matched to signify that elusive "clean house" — you can keep the kids out of the house longer by increasing the number in line 1420. Then enjoy the fantasy of a completely clean house all the time. What? Why can't your home computer make this fantasy actually come true? Be patient. It's just a matter of time. . . Anyway, in the words of a once-popular song: "Such are the dreams of the everyday housewife. . ."

Programming Techniques

This program illustrates the capabilities of TI-99/4A color graphics. Characters are defined in each of the eight user-defined character sets, and each set has a different color scheme. These eight sets are used for the eight chores; and for ease in programming, they are numbered 1 through 8.

Two characters in Set 2 are also redefined with a blue foreground and a red background ("FFFFFFFFFFFFFFFF" and "0") to draw the 16-square checkerboard array. It is drawn with a triple-nested FOR-NEXT loop (statements 2040-2150).

The eight chores to be drawn are called in subroutines (statements 2290 to 3060). The subroutines use x- and y-coordinates to define the placement of the special characters. The coordinates are specified before the subroutine is called. The coordinates of the chores for each of the sixteen squares are listed in subroutines also (statements 5350-5980).

To set up the array of 16 squares, two arrays are actually used: WORK(16) and HH(16). The WORK array is given the numbers of the eight chores: WORK(1) = 1; WORK(2) = 2; . . . WORK(9) = 1; WORK(10) = 2; and so on (statements 3370-3400). For the HH array, a subscript RR is chosen as a random number from 1 to 16. HH(RR) is then set equal to WORK(RR), and then WORK(RR) = 0 so it won't be chosen again. This process continues until all 16 numbers of the HH array have been filled randomly with the numbers from the WORK array (statements 3410-3470). These numbers are the chore numbers for the squares. For example, HH(4) = 7 means that behind the 4th square (D) would be chore number 7 (shopping).

The WORK array is then reset equal to the HH array so the chores can be printed in order on the squares for a "clean house" or for "stop".

As the game is being played, the HH elements are set equal to zero if a match is made, so the match can only be scored once. If a player chooses a square which has previously been part of a matched pair, the word "DONE" appears across the square.

EXPLANATION OF THE PROGRAM
Harried Housewife

Line Nos.

150-180	Prints title screen.
190-260	Defines colors for eight household chores.
270-820	Defines special characters for drawing the chores.
830	Displays the eight chores on title screen.
840-850	Sets counters for the number of trial guesses and the number of successful matches.
860	Dimensions arrays to handle 16 elements.
870-880	Redefines characters for checkerboard.
890-900	Delays for title screen.
910-920	Clears screen and makes it yellow.
930	Defines colors for checkerboard.
940	Draws checkerboard and labels it.
950	Assigns the chores for each square in array.
960-1010	Prints HOUSEWORK.
1020-1130	Prints MATCH 2 LETTERS.
1140-1160	Prints two red lines for the letters chosen.
1170-1200	Waits for letter A-P to be pressed.
1210	Prints the chosen letter.
1220-1230	Finds chore number and coordinates for square chosen.
1240-1260	If the square has been previously matched, prints DONE.
1270	Draws first chore on square.
1280-1330	Waits for second letter to be pressed and prints it.
1340-1350	Finds the chore number and coordinates for that square.
1360-1390	Prints DONE.
1400	Draws second chore on square.
1410	Checks for a match.
1420	Increments the number of trials.
1430	If TIME = 10 prints message to hurry.
1440	If TIME = 12, kids come home.
1450	Branches if TIME is less than 10.
1460	Clears previous message.
1470-1520	Prints OH NO! KIDS ARE HOME!
1530-1550	Reprints checkerboard and scrambles chores for a new game.
1560	Prints PRESS ENTER TO CONTINUE and waits for response, covers squares for next choice.
1570-1620	Prints SPEED-KIDS WILL BE HOME SOON!
1630	Same as 1540
1640-1750	Correct match is made, sounds tone of A, prints finished chore.
1760-1770	Sets elements matched to zero so they can't be scored again.
1780	Returns for next choice.
1790-1840	If all eight matches have been made, prints CLEAN HOUSE!!
1850	Prints S if player wants to stop.
1860	Resets HH array to current arrangement.
1870-1910	Shows all chores in array.
1920-1980	Clears all other printing.
1990-2040	Prints HOUSEWORK NEVER ENDS.
2050	Holds screen until SHIFT C (BREAK) is pressed.

Subroutines:

2060-2170	Prints checkerboard.
2180-2230	Prints letters A to P on squares.
2240-2300	Prints S = STOP and returns.
2310-2410	Draws feather duster.
2420-2490	Draws sewing machine.
2500-2600	Draws T-shirt for washing.
2620-2690	Draws cup and saucer for dishes.
2700-2780	Draws pan for cooking.
2790-2880	Draws vacuum cleaner.
2890-2980	Draws shopping basket.
2990-3080	Draws ironing board.
3090-3300	Places symbols on title screen.
3310-3380	Plays music for title screen.
3390-3420	Puts two sets of chore numbers in WORK array.
3430-3490	Randomly arranges chores in HH array, two of each chore.
3500-3520	Resets WORK array equal to HH array.
3530	Restarts number of matches.
3540-3580	Clears printed list of matches made.
3590-3620	Resets HH array to original WORK array for printing.
3630-4490	When a match is made, blinks the picture and prints the chore in the "Finished" list; prints labels under pictures in the squares.
4500-4570	Prints PRESS ENTER TO CONTINUE and waits for response.
4580-4590	Clears messages.
4600-4630	Covers squares again and relabels them.
4640	Return for next choice.
4650-5280	Subroutines for covering particular square.
5290-5320	Colors blue square.
5330-5360	Colors red square.
5370-6000	Designates the chore number and coordinates for the square chosen.

```
100 REM *********************
110 REM * MARRIED HOUSEWIFE *
120 REM *********************
130 REM
140 REM
150 CALL CLEAR
160 PRINT TAB(10);"MARRIED"
170 PRINT ::TAB(9);"HOUSEWIFE"
180 PRINT ::::::::::
190 CALL COLOR(9,7,15)
200 CALL COLOR(10,15,12)
210 CALL COLOR(11,14,11)
220 CALL COLOR(12,16,3)
230 CALL COLOR(13,7,12)
240 CALL COLOR(14,5,8)
250 CALL COLOR(15,15,16)
260 CALL COLOR(16,3,16)
270 CALL CHAR(96,"00000040E8EBEFFFF")
280 CALL CHAR(97,"00000000020E0C0C")
290 CALL CHAR(98,"0201010703010101")
300 CALL CHAR(99,"FFFFFFFFFFFFFFFF")
310 CALL CHAR(100,"F0E0C0F0F1E080C")
320 CALL CHAR(101,"03070E1C3870E0C")
330 CALL CHAR(102,"FF0E03")
340 CALL CHAR(103,"0")
350 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
360 CALL CHAR(105,"FCFCFCFFFFFFFFFF")
370 CALL CHAR(106,"FCFCF8F8F0E0C")
380 CALL CHAR(107,"FFFFFFFFFCFCFCFC")
390 CALL CHAR(108,"FFFFFFFFFF")
400 CALL CHAR(109,"0")
410 CALL CHAR(112,"000000000F0F0F0F")
420 CALL CHAR(113,"0000000081C3FFFF")
430 CALL CHAR(114,"00000000F0F0F0F0")
440 CALL CHAR(115,"0F0F")
450 CALL CHAR(116,"FFFFFFFFFFFFFFFF")
460 CALL CHAR(117,"F0F")
470 CALL CHAR(118,"FFFFFFFF")
480 CALL CHAR(119,"0")
490 CALL CHAR(120,"000000000000C0F0F")
500 CALL CHAR(121,"000000000000FFFF")
510 CALL CHAR(122,"000000000000C0C0FC")
520 CALL CHAR(123,"0F0F0F0F0F0F0F07")
530 CALL CHAR(124,"FFFFFFFFFFFFFFFF")
540 CALL CHAR(125,"FEC6C6C6DCF8E00")
550 CALL CHAR(126,"FFFF")
560 CALL CHAR(128,"1F1F1F1F1F1F1F1F")
570 CALL CHAR(129,"FFFFFCFCFCFCFCFC")
580 CALL CHAR(130,"FFFF")
590 CALL CHAR(131,"1F1F0F")
600 CALL CHAR(132,"FCFCF8")
610 CALL CHAR(133,"0")
620 CALL CHAR(136,"1F0F01000000000303")
630 CALL CHAR(137,"80C0C0C0C0F0F8F8")
640 CALL CHAR(138,"0303030303030101")
650 CALL CHAR(139,"F8F0F0F0F0F0E0E")
660 CALL CHAR(140,"E0C7CF7FFFFFFFFF")
670 CALL CHAR(141,"0080C0C0F0F0E")
680 CALL CHAR(142,"0")
690 CALL CHAR(144,"000000000040C1A19")
700 CALL CHAR(145,"000F09090F09090F")
710 CALL CHAR(146,"FE252424FF2424FF")
720 CALL CHAR(147,"00E09E92FE9292FE")
730 CALL CHAR(148,"06090906")
740 CALL CHAR(149,"0")
750 CALL CHAR(152,"000000001F1F1F1F")
760 CALL CHAR(153,"00000000F0FFFFFF")
770 CALL CHAR(154,"0000000000F0FEFE")
780 CALL CHAR(155,"1F1F1F040201")
790 CALL CHAR(156,"FFFFE2040810A04")
800 CALL CHAR(157,"090A0C0808")
810 CALL CHAR(158,"A1190701010")
820 CALL CHAR(159,"0")
830 GOSUB 3000
840 TIME=0
850 MATCH=0
860 DIM HH(16),WORK(16)
870 CALL CHAR(45,"FFFFFFFFFFFFFFFF")
880 CALL CHAR(44,"0")
890 CALL SOUND(4225,44000,30)
900 CALL SOUND(4,44000,30)
910 CALL CLEAR
920 CALL SCREEN(12)
930 CALL COLOR(2,6,9)
940 GOSUB 2060
950 GOSUB 3390
960 DATA 72,79,85,83,69,87,79,82,75
970 RESTORE 960
980 FOR Y=23 TO 31
990 READ GR
1000 CALL HCHAR(2,Y,GR)
1010 NEXT Y
1020 DATA 77,65,84,67,72,32,50
1030 RESTORE 1020
1040 FOR Y=23 TO 29
1050 READ GR
1060 CALL HCHAR(5,Y,GR)
1070 NEXT Y
1080 DATA 76,69,84,84,69,82,83
1090 RESTORE 1080
1100 FOR Y=23 TO 29
1110 READ GR
1120 CALL HCHAR(6,Y,GR)
1130 NEXT Y
1140 CALL COLOR(8,7,1)
1150 CALL HCHAR(8,25,95)
1160 CALL HCHAR(8,27,95)
1170 CALL KEY(0,K1,ST)
1180 IF K1=83 THEN 1850
1190 IF K1<65 THEN 1170
1200 IF K1>80 THEN 1170
1210 CALL HCHAR(8,25,K1)
1220 SS=1
1230 ON (K1-64)GOSUB 5370,5410,5450,549
     0,5530,5570,5610,5650,5690,5730,57
     70,5810,5850,5890,5930,5970
1240 IF CH(1)<>0 THEN 1270
1250 GOSUB 4450
1260 GOTO 1280
1270 ON CH(1)GOSUB 2310,2420,2500,2610,
     2700,2790,2890,2990
1280 CALL KEY(0,K2,ST)
1290 IF K2=83 THEN 1850
1300 IF K2<65 THEN 1280
1310 IF K2>80 THEN 1280
1320 IF K2=K1 THEN 1280
1330 CALL HCHAR(8,27,K2)
1340 SS=2
1350 ON (K2-64)GOSUB 5370,5410,5450,549
     0,5530,5570,5610,5650,5690,5730,57
     70,5810,5850,5890,5930,5970
1360 IF CH(2)<>0 THEN 1400
1370 GOSUB 4450
1380 GOTO 1420
1390 IF CH(1)=0 THEN 1420
1400 ON CH(2)GOSUB 2310,2420,2500,2610,
     2700,2790,2890,2990
1410 IF CH(1)=CH(2)THEN 1640
1420 TIME=TIME+1
1430 IF TIME=10 THEN 1570
1440 IF TIME=12 THEN 1460
1450 GOTO 4500
1460 CALL HCHAR(22,2,32,31)
1470 DATA 79,72,32,78,79,33,32,75,73,68
     ,83,32,65,82,69,32,72,79,77,69,33
1480 RESTORE 1470
1490 FOR Y=3 TO 23
1500 READ GR
1510 CALL HCHAR(24,Y,GR)
1520 NEXT Y
1530 GOSUB 2060
1540 GOSUB 3390
1550 TIME=0
1560 GOTO 4500
1570 DATA 83,80,69,69,68,59,32,75,73,68
     ,83,32,87,73,76,76,32,66,69,32,72,
     79,77,69,32,83,79,79,78,33
1580 RESTORE 1570
```

```
1590 FOR Y=2 TO 31
1600 READ GR
1610 CALL HCHAR(22,Y,GR)
1620 NEXT Y
1630 GOTO 4500
1640 CALL SOUND(1000,440,2)
1650 MATCH=MATCH+1
1660 IF MATCH<>1 THEN 1730
1670 DATA 70,73,78,73,83,72,69,68,58
1680 RESTORE 1670
1690 FOR Y=23 TO 31
1700 READ GR
1710 CALL HCHAR(11,Y,GR)
1720 NEXT Y
1730 X=MATCH+9
1740 Y=26
1750 ON CH(1)GOSUB 3630,3720,3800,3890,
     4050,4130,4270,4360
1760 HH(X1-64)=0
1770 HH(X2-64)=0
1780 IF MATCH<>8 THEN 1420
1790 DATA 67,76,69,65,78,32,72,79,85,83
     ,69,33,33
1800 RESTORE 1790
1810 FOR Y=5 TO 27 STEP 2
1820 READ GR
1830 CALL HCHAR(24,Y,GR)
1840 NEXT Y
1850 CALL HCHAR(8,25,83)
1860 GOSUB 3590
1870 FOR S=1 TO 16
1880 SS=S
1890 ON S GOSUB 5370,5410,5450,5490,553
     0,5570,5610,5650,5690,5730,5770,58
     10,5850,5890,5930,5970
1900 ON CH(SS)GOSUB 2510,2420,2500,2610
     ,2700,2790,2880,2900
1910 NEXT S
1920 CALL HCHAR(21,3,32,6)
1930 CALL HCHAR(22,2,32,31)
1940 IF MATCH<>8 THEN 1990
1950 FOR X=2 TO 8
1960 CALL HCHAR(X,23,32,9)
1970 NEXT X
1980 GOTO 2050
1990 DATA 72,79,85,83,69,87,79,82,75,32
     ,78,69,86,69,82,32,69,78,68,33,33
2000 RESTORE 1990
2010 FOR Y=3 TO 23
2020 READ GR
2030 CALL HCHAR(24,Y,GR)
2040 NEXT Y
2050 GOTO 2050
2060 FOR Z=1 TO 11 STEP 10
2070 FOR X=Z TO Z+4
2080 FOR Y=2 TO 12 STEP 10
2090 CALL SOUND(100,1047,2)
2100 CALL HCHAR(X,Y,43,5)
2110 CALL HCHAR(X,Y+5,44,5)
2120 CALL SOUND(100,523,2)
2130 CALL HCHAR(X+5,Y,44,5)
2140 CALL HCHAR(X+5,Y+5,43,5)
2150 NEXT Y
2160 NEXT X
2170 NEXT Z
2180 DATA 3,4,3,3,3,14,3,10,8,4,8,9,8,1
     4,8,19,13,4,13,9,13,14,13,19,18,4,
     18,9,18,14,18,19
2190 RESTORE 2180
2200 FOR CC=65 TO 80
2210 READ X,Y
2220 CALL HCHAR(X,Y,CC)
2230 NEXT CC
2240 CALL HCHAR(21,3,83)
2250 CALL HCHAR(21,4,61)
2260 CALL HCHAR(21,5,83)
2270 CALL HCHAR(21,6,84)
2280 CALL HCHAR(21,7,79)
2290 CALL HCHAR(21,8,80)
2500 RETURN
2510 CALL HCHAR(X-1,Y,96)
2520 CALL HCHAR(X-1,Y-1,103)
2530 CALL HCHAR(X-1,Y+1,97)
2540 CALL HCHAR(X,Y-1,98)
2550 CALL HCHAR(X,Y,99)
2560 CALL HCHAR(X,Y+1,100)
2570 CALL HCHAR(X+1,Y-1,101)
2580 CALL HCHAR(X+1,Y,102)
2590 CALL HCHAR(X+1,Y+1,103)
2600 GOSUB 3670
2610 RETURN
2620 CALL HCHAR(X-1,Y-1,104,2)
2630 CALL HCHAR(X,Y,109)
2640 CALL HCHAR(X-1,Y+1,105)
2650 CALL HCHAR(X,Y-1,106)
2660 CALL HCHAR(X,Y+1,107)
2670 CALL HCHAR(X+1,Y-1,108,3)
2680 GOSUB 3700
2690 RETURN
2700 CALL HCHAR(X-1,Y-1,112)
2710 CALL HCHAR(X-1,Y,113)
2720 CALL HCHAR(X-1,Y+1,114)
2730 CALL HCHAR(X,Y-1,115)
2740 CALL HCHAR(X,Y,116)
2750 CALL HCHAR(X,Y+1,117)
2760 CALL HCHAR(X+1,Y-1,118)
2770 CALL HCHAR(X+1,Y,119)
2780 CALL HCHAR(X+1,Y+1,119)
2790 GOSUB 3840
2800 RETURN
2810 CALL HCHAR(X-1,Y-1,120)
2820 CALL HCHAR(X-1,Y,121)
2830 CALL HCHAR(X-1,Y+1,122)
2840 CALL HCHAR(X,Y-1,123)
2850 CALL HCHAR(X,Y,124)
2860 CALL HCHAR(X,Y+1,125)
2870 CALL HCHAR(X+1,Y-1,126,3)
2880 GOSUB 4000
2890 RETURN
2900 CALL HCHAR(X,Y-1,128)
2910 CALL HCHAR(X-1,Y-1,135,3)
2920 CALL HCHAR(X+1,Y+1,133)
2930 CALL HCHAR(X,Y,129)
2940 CALL HCHAR(X,Y+1,130)
2950 CALL HCHAR(X+1,Y-1,131)
2960 CALL HCHAR(X+1,Y,132)
2970 GOSUB 4099
2980 RETURN
2990 CALL HCHAR(X-1,Y-1,136)
3000 CALL VCHAR(X-1,Y+1,142,2)
3010 CALL HCHAR(X+1,Y-1,142)
3020 CALL HCHAR(X-1,Y,137)
3030 CALL HCHAR(X,Y-1,138)
3040 CALL HCHAR(X,Y,139)
3050 CALL HCHAR(X+1,Y,140)
3060 CALL HCHAR(X+1,Y+1,141)
3070 GOSUB 4230
3080 RETURN
3090 CALL HCHAR(X-1,Y-1,144)
3100 CALL HCHAR(X-1,Y,145,2)
3110 CALL HCHAR(X+1,Y,145)
3120 CALL HCHAR(X,Y-1,145)
3130 CALL HCHAR(X,Y,146)
3140 CALL HCHAR(X,Y+1,147)
3150 CALL HCHAR(X+1,Y-1,148)
3160 CALL HCHAR(X+1,Y+1,148)
3170 GOSUB 4310
3180 RETURN
3190 CALL HCHAR(X-1,Y-1,152)
3200 CALL HCHAR(X-1,Y,153)
3210 CALL HCHAR(X-1,Y+1,154)
3220 CALL HCHAR(X,Y-1,155)
3230 CALL HCHAR(X,Y,156)
3240 CALL HCHAR(X+1,Y-1,157)
3250 CALL HCHAR(X+1,Y,158)
3260 CALL VCHAR(X,Y+1,159,2)
3270 GOSUB 4400
3280 RETURN
```

```
3090 X=3
3100 Y=5
3110 GOSUB 2310
3120 X=4
3130 Y=16
3140 GOSUB 2420
3150 Y=27
3160 GOSUB 2500
3170 X=8
3180 Y=7
3190 GOSUB 2610
3200 X=16
3210 Y=26
3220 GOSUB 2700
3230 X=17
3240 GOSUB 2790
3250 X=16
3260 Y=15
3270 GOSUB 2890
3280 X=15
3290 Y=6
3300 GOSUB 2990
3310 CALL SOUND(300,494.2,196.7)
3320 CALL SOUND(200,440.2)
3330 CALL SOUND(200,392.2)
3340 CALL SOUND(300,440.2,185.8)
3350 CALL SOUND(200,392.3)
3360 CALL SOUND(200,370.3)
3370 CALL SOUND(1000,392.3,165.8)
3380 RETURN
3390 FOR Z=1 TO 8
3400 WORK(Z)=Z
3410 WORK(Z+8)=Z
3420 NEXT Z
3430 RANDOMIZE
3440 FOR R=1 TO 16
3450 RR=INT(16*RND)+1
3460 IF WORK(RR)=0 THEN 3450
3470 HH(R)=WORK(RR)
3480 WORK(RR)=0
3490 NEXT R
3500 FOR R=1 TO 16
3510 WORK(R)=HH(R)
3520 NEXT R
3530 MATCH=0
3540 FOR X=11 TO 18
3550 CALL HCHAR(X,25,32,9)
3560 NEXT X
3570 CALL HCHAR(24,3,32,22)
3580 RETURN
3590 FOR R=1 TO 16
3600 HH(R)=WORK(R)
3610 NEXT R
3620 RETURN
3630 CALL COLOR(9,15,7)
3640 CALL COLOR(9,7,15)
3650 CALL COLOR(9,15,7)
3660 CALL COLOR(9,7,15)
3670 CALL HCHAR(X+2,Y-1,68)
3680 CALL HCHAR(X+2,Y,85)
3690 CALL HCHAR(X+2,Y+1,83)
3700 CALL HCHAR(X+2,Y+2,84)
3710 RETURN
3720 CALL COLOR(10,12,13)
3730 CALL COLOR(10,13,12)
3740 CALL COLOR(10,12,13)
3750 CALL COLOR(10,13,12)
3760 CALL HCHAR(X+2,Y-1,83)
3770 CALL HCHAR(X+2,Y,69)
3780 CALL HCHAR(X+2,Y+1,87)
3790 RETURN
3800 CALL COLOR(11,11,14)
3810 CALL COLOR(11,14,11)
3820 CALL COLOR(11,11,14)
3830 CALL COLOR(11,14,11)
3840 CALL HCHAR(X+2,Y-1,87)
3850 CALL HCHAR(X+2,Y,65)
3860 CALL HCHAR(X+2,Y+1,83)
3870 CALL HCHAR(X+2,Y+2,72)
3880 RETURN
3890 CALL COLOR(12,3,16)
3900 CALL COLOR(12,16,5)
3910 CALL COLOR(12,5,16)
3920 CALL COLOR(12,16,3)
3930 CALL HCHAR(X+2,25,68)
3940 CALL HCHAR(X+2,26,75)
3950 CALL HCHAR(X+2,27,83)
3960 CALL HCHAR(X+2,28,72)
3970 CALL HCHAR(X+2,29,69)
3980 CALL HCHAR(X+2,30,83)
3990 RETURN
4000 CALL HCHAR(X+2,Y-1,68)
4010 CALL HCHAR(X+2,Y,73)
4020 CALL HCHAR(X+2,Y+1,83)
4030 CALL HCHAR(X+2,Y+2,72)
4040 RETURN
4050 CALL COLOR(13,12,7)
4060 CALL COLOR(13,7,12)
4070 CALL COLOR(13,12,7)
4080 CALL COLOR(13,7,12)
4090 CALL HCHAR(X+2,Y-1,67)
4100 CALL HCHAR(X+2,Y,79,2)
4110 CALL HCHAR(X+2,Y+2,75)
4120 RETURN
4130 CALL COLOR(14,8,5)
4140 CALL COLOR(14,5,8)
4150 CALL COLOR(14,8,5)
4160 CALL COLOR(14,5,8)
4170 CALL HCHAR(X+2,25,86)
4180 CALL HCHAR(X+2,26,65)
4190 CALL HCHAR(X+2,27,67)
4200 CALL HCHAR(X+2,28,85,2)
4210 CALL HCHAR(X+2,30,77)
4220 RETURN
4230 CALL HCHAR(X+2,Y-1,86)
4240 CALL HCHAR(X+2,Y,65)
4250 CALL HCHAR(X+2,Y+1,67)
4260 RETURN
4270 CALL COLOR(15,16,15)
4280 CALL COLOR(15,15,16)
4290 CALL COLOR(15,16,15)
4300 CALL COLOR(15,15,16)
4310 CALL HCHAR(X+2,Y-1,85)
4320 CALL HCHAR(X+2,Y,72)
4330 CALL HCHAR(X+2,Y+1,79)
4340 CALL HCHAR(X+2,Y+2,80)
4350 RETURN
4360 CALL COLOR(16,16,3)
4370 CALL COLOR(16,3,16)
4380 CALL COLOR(16,16,3)
4390 CALL COLOR(16,3,16)
4400 CALL HCHAR(X+2,Y-1,73)
4410 CALL HCHAR(X+2,Y,82)
4420 CALL HCHAR(X+2,Y+1,79)
4430 CALL HCHAR(X+2,Y+2,78)
4440 RETURN
4450 CALL HCHAR(X,Y-1,68)
4460 CALL HCHAR(X,Y,79)
4470 CALL HCHAR(X,Y+1,78)
4480 CALL HCHAR(X,Y+2,69)
4490 RETURN
4500 DATA 89,82,69,85,83,32,69,78,84,69
     ,82,32,84,79,32,67,79,78,84,73,78,
     85,69,32
4510 RESTORE 4500
4520 FOR Y=3 TO 26
4530 READ GR
4540 CALL HCHAR(23,Y,GR)
4550 NEXT Y
4560 CALL KEY(0,KEY,ST)
4570 IF KEY<>13 THEN 4560
4580 CALL HCHAR(4,24,32,5)
4590 CALL HCHAR(23,2,32,25)
4600 ON (K1-64) GOSUB 4650,4690,4730,477
     0,4810,4850,4890,4930,4970,5010,50
     50,5090,5130,5170,5210,5250
```

```
 4610  CALL HCHAR(X,Y,E1)
 4620  ON (E2-64)GOSUB 4650,4690,4730,477
       0,4810,4850,4890,4930,4970,5010,50
       50,5090,5130,5170,5210,5250
 4630  CALL HCHAR(X,Y,E2)
 4640  GOTO 1150
 4650  X=5
 4660  Y=4
 4670  GOSUB 5290
 4680  RETURN
 4690  X=5
 4700  Y=9
 4710  GOSUB 5330
 4720  RETURN
 4730  X=5
 4740  Y=14
 4750  GOSUB 5290
 4760  RETURN
 4770  X=5
 4780  Y=19
 4790  GOSUB 5330
 4800  RETURN
 4810  X=8
 4820  Y=4
 4830  GOSUB 5330
 4840  RETURN
 4850  X=8
 4860  Y=9
 4870  GOSUB 5290
 4880  RETURN
 4890  X=8
 4900  Y=14
 4910  GOSUB 5330
 4920  RETURN
 4930  X=8
 4940  Y=19
 4950  GOSUB 5290
 4960  RETURN
 4970  X=13
 4980  Y=4
 4990  GOSUB 5290
 5000  RETURN
 5010  X=13
 5020  Y=9
 5030  GOSUB 5330
 5040  RETURN
 5050  X=13
 5060  Y=14
 5070  GOSUB 5290
 5080  RETURN
 5090  X=13
 5100  Y=19
 5110  GOSUB 5330
 5120  RETURN
 5130  X=18
 5140  Y=4
 5150  GOSUB 5330
 5160  RETURN
 5170  X=18
 5180  Y=9
 5190  GOSUB 5290
 5200  RETURN
 5210  X=18
 5220  Y=14
 5230  GOSUB 5330
 5240  RETURN
 5250  X=18
 5260  Y=19
 5270  GOSUB 5290
 5280  RETURN
 5290  FOR XX=X-1 TO X+2
 5300  CALL HCHAR(XX,Y-1,43,4)
 5310  NEXT XX
 5320  RETURN
 5330  FOR XX=X-1 TO X+2
 5340  CALL HCHAR(XX,Y-1,44,4)
 5350  NEXT XX
 5360  RETURN
 5370  CH(SS)=HH(1)
 5380  X=3
 5390  Y=4
 5400  RETURN
 5410  CH(SS)=HH(2)
 5420  X=5
 5430  Y=9
 5440  RETURN
 5450  CH(SS)=HH(3)
 5460  X=3
 5470  Y=14
 5480  RETURN
 5490  CH(SS)=HH(4)
 5500  X=3
 5510  Y=19
 5520  RETURN
 5530  CH(SS)=HH(5)
 5540  X=8
 5550  Y=4
 5560  RETURN
 5570  CH(SS)=HH(6)
 5580  X=8
 5590  Y=9
 5600  RETURN
 5610  CH(SS)=HH(7)
 5620  X=8
 5630  Y=14
 5640  RETURN
 5650  CH(SS)=HH(8)
 5660  X=8
 5670  Y=19
 5680  RETURN
 5690  CH(SS)=HH(9)
 5700  X=13
 5710  Y=4
 5720  RETURN
 5730  CH(SS)=HH(10)
 5740  X=13
 5750  Y=9
 5760  RETURN
 5770  CH(SS)=HH(11)
 5780  X=13
 5790  Y=14
 5800  RETURN
 5810  CH(SS)=HH(12)
 5820  X=13
 5830  Y=19
 5840  RETURN
 5850  CH(SS)=HH(13)
 5860  X=18
 5870  Y=4
 5880  RETURN
 5890  CH(SS)=HH(14)
 5900  X=18
 5910  Y=9
 5920  RETURN
 5930  CH(SS)=HH(15)
 5940  X=18
 5950  Y=14
 5960  RETURN
 5970  CH(SS)=HH(16)
 5980  X=18
 5990  Y=19
 6000  RETURN
```

END

EXTENDED BASIC

FORCE 1

You are the Captain of the Force 1, a United Federation of Planets police cruiser. A message has just come in that a large number of alien bandits have ntered your sector and are planning an attack on your home lanet. The bandits cannot be taken alive and therefore must e destroyed. The job won't be easy, so you'd better stay lert.

Since the bandits are armed with short-range laser canons, they should be encountered when beyond their firing ange. As you become a better pilot, you may choose to ncrease your ship's speed with higher levels of difficulty. This means that the alien craft will be approached much nore rapidly, and more accuracy on your part is needed.

On first sighting, your radar screen will show the alien o be no larger than the background stars, and very difficult to pick out among them. As you approach the ship, will become larger and larger, until the alien is either in ange to fire its laser cannon, or slightly out of range flying ight past you.

To maneuver your ship in order to set your gun sights n the alien bandit, you must use the four arrow keys. If ou hold a key down continually, your ship will keep aelerating in that direction. This will, of course, cause the ar field and the alien ship to move more in the opposite irection. For example, if the alien ship were moving off the right of the screen and you wanted to bring him back to the center, you would hold the D key down until the ien started moving toward the center. Then to halt all ovement by the alien and keep him from going to the left f the screen, press the S key until the alien either stops or ows down to a minimum speed. The idea is to slow his orizontal and vertical speed to a minimum and position m in the center of your gun sight. To fire your laser blaster, ess ENTER. Getting the alien in your gun sight may not as easy as it sounds, for the alien is intelligent and eriodically shifts course like all skillful space bandits. So st when you think you have him, he's off in another rection . . .

You have 1000 units of time to complete your mission before the strike on your planet. If 25 or more bandit ships are destroyed, you will gain an extra 1000 units of time to attack the second wave of aliens.

The Program

The program is written in Extended BASIC. I decided here to make use of the MAGNIFY commands to create a series of space ships that start off very small and gradually become larger. This gives a more realistic view of an object coming closer. I gave the ship a random speed—slow at first when it's at a great distance, and accelerating as it gets closer. I also gave the ship the ability to change directions randomly 10 percent of the time. The ability to use sprites for both the ship and the star field made it possible to create the illusion of actual motion—not just changing the alien's direction in reference to yours, but also with respect to every star in the star field. For example, take the case of the alien ship traveling to the right of the screen and all of the stars not moving. If you press the D key until the alien stops moving, all of the stars will now be moving to the left, and the alien will be still. This works the same way vertically.

By using the COINCidence statement and the tolerance option, I was able to make it more difficult to hit a ship at a greater distance (where it needs to be a direct hit) than to hit one that is nearby. There is however a slight time delay from the time you press the {ENTER} key until the laser fires. This makes it almost impossible to hit a moving target. So the challenge will be to get the alien in your gun sights and hold him there long enough to make a successful strike.

The laser bolts that you fire at the alien are there all of the time, but kept invisible. I then use the CALL COLOR statement twice—once to turn on the bolts, and once to turn them back off.

If the alien ship is still in your gun sight when it reaches maximum size, you will be within range of his laser cannon and be fired upon. WARNING: Laser cannons never miss at short range!

EXPLANATION OF THE PROGRAM

Force 1

Line Nos.	
130-210	Display levels of difficulty; accept answer.
220-460	Assign variables, color, and characters.
470-560	Read keyboard, branch to subroutine, or adjust variables.
570-610	Adjust distance to alien; branch to display new alien ship.
620-630	Randomly change motion of alien ship.
640-650	Change motion of stars.
660-670	Display score, time; check for out of time (time = 1000).
680-830	Display laser beams on screen.
840-860	Assign alien space craft to a new location.
870-880	Fire laser, check for hit.
890-910	Alien destroyed. Adjust score, re-initialize variables.
920-1060	Subroutine when hit by alien; branch for bonus, or branch to end-of-game messages.
1070-1180	End-of-game messages.
1190-1210	Check to play again.
1220-1250	Change alien shape.
1260-1270	Check for alien to fire back.
1280-1340	Alien is at maximum size and moves off screen faster.
1350-1420	Alien fires and hits your ship; sound effects.
1430-1520	Display star pattern.
1530-1630	Display title page.

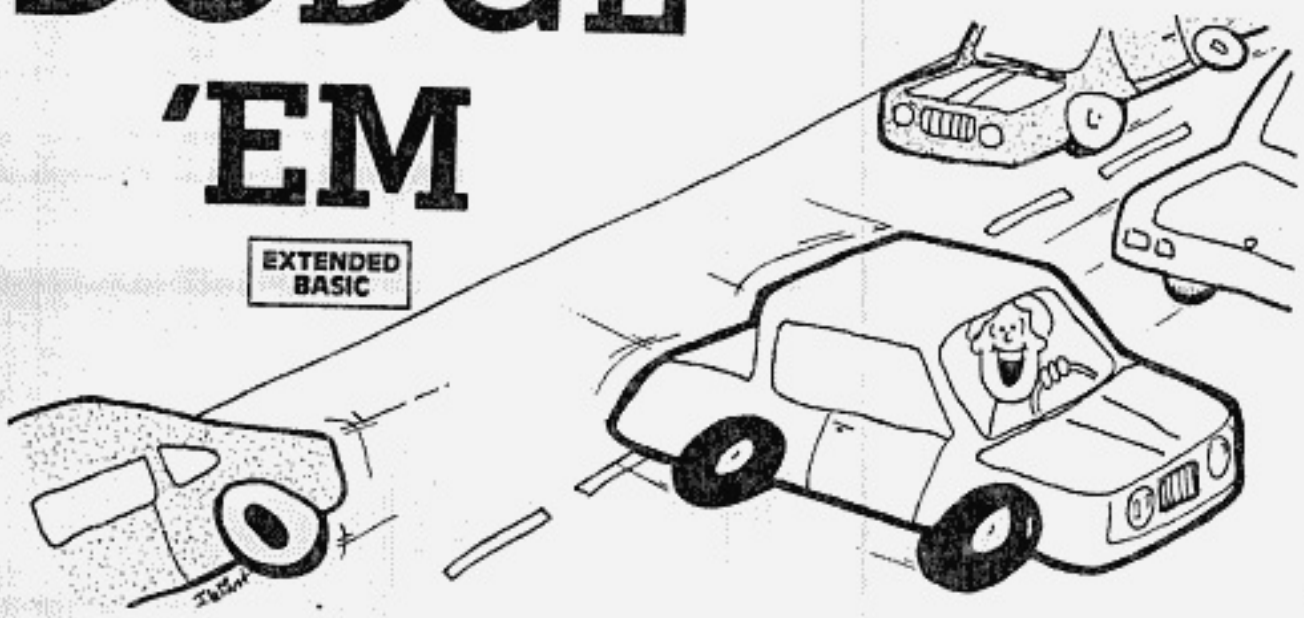
```
100 REM *********
110 REM * FORCE 1 *
120 REM *********
130 CALL CLEAR :: GOSUB 1530
140 DISPLAY AT(2,10):"FORCE 1"
150 DISPLAY AT(4,5):"LEVEL OF DIFFICUL
TY:"
160 DISPLAY AT(6,5):"1. BEGINNER" :: D
ISPLAY AT(7,5):"2. NOVICE" :: DISP
LAY AT(8,5):"3. INTERMEDIATE"
170 DISPLAY AT(9,5):"4. SEMI-PRO" :: D
ISPLAY AT(10,5):"5. PRO"
180 ACCEPT AT(11,5)VALIDATE(DIGIT)SIZE
(1):L1 :: L=L1+4
190 RANDOMIZE :: CALL CLEAR
200 FOR CO=1 TO 8 :: CALL COLOR(CO,16,
1):: NEXT CO
210 COU=0 :: D=1 :: DIS=11000 :: IF SC
>=25 THEN L=L+2
220 CALL CHAR(88,"0102040810204080"):: :
CALL CHAR(89,"8040201008040201")
230 CALL CHAR(90,"03070E1C3870E0C0"):: :
CALL CHAR(91,"C0E070381C0E0703")
240 CALL CHAR(92,"0707F1F3E7CF8F0E0"):: :
CALL CHAR(93,"E0F0F8F7C3E1F0F07")
250 CALL CHAR(94,"8381860C183061C0"):: :
CALL CHAR(95,"C061830186C08301")
260 CALL COLOR(8,1,1):: CALL SCREEN(2)
270 CALL CHAR(96,"0101010101010101"):: :
CALL CHAR(97,"8080808080808080"):: :
280 CALL CHAR(98,"00000000000000FF"):: :
CALL CHAR(99,"FF")
290 CALL COLOR(9,16,1)
300 CALL VCHAR(7,12,96,9):: CALL VCHAR
(7,21,87,9)
310 CALL HCHAR(6,13,98,8):: CALL HCHAR
(16,13,99,8)
320 CALL CHAR(33,"FF"):: CALL CHAR(34,
"0101010101010101"):: :
330 CALL VCHAR(12,15,35):: CALL VCHAR(
12,18,33):: CALL VCHAR(10,16,34):: :
CALL VCHAR(13,16,34)
340 FOR COL=10 TO 12 :: CALL COLOR(COL
,7,1):: NEXT COL
350 GOSUB 360 :: GOTO 450
360 CALL CHAR(104,"0000000018"):: CALL C
HAR(105,"0000001818"):: CALL CHAR(10
6,"000000183C"):: :
370 CALL CHAR(107,"00000018183C"):: CALL C
HAR(108,"0000183C3C"):: CALL CHAR(10
9,"00001C3E3E"):: :
380 CALL CHAR(110,"00005C7E18"):: CALL
CHAR(111,"001878FFFF5C4200"):: :
390 CALL CHAR(112,"003C1E7FFF5F4000"):: :
CALL CHAR(113,"00"):: CALL CHAR(114
,"0000000000C00300"):: CALL CHAR(115
,"00"):: :
400 CALL CHAR(116,"00000000061F7FFFFFFF"):: :
CALL CHAR(117,"3F2040"):: CALL C
HAR(118,"0000000000891F0F0F"):: CALL
CHAR(119,"C04020"):: :
410 CALL CHAR(120,"0000000000107FFF"):: :
CALL CHAR(121,"FF1F133060400"):: :
CALL CHAR(122,"0000000000801F0FCFF"):: :
420 CALL CHAR(123,"FFF8F880C0602"):: :
430 CALL CHAR(124,"02604CD700330C01"):: :
440 RETURN
450 CALL COLOR(12,7,1)
460 GOSUB 690 :: CALL MAGNIFY(1):: GOS
UB 840 :: GOSUB 1410
470 CALL KEY(0,K,S)
480 CALL POSITION(#1,P01,P02)
490 IF K=13 THEN GOSUB 870
500 T=INT(RND*10):: IF T=4 THEN DEV=1/
10-INT(RND*1/5):: DEV=1/10-INT(RND
*1/5)ELSE DEV=DEV*0
510 IF K=69 THEN D1=D1+1/5 :: SA=SA+1/
5 :: IF SA>127 THEN SA=127
520 IF K=88 THEN D1=D1-1/5 :: SA=SA-1/
5 :: IF SA<-128 THEN SA=-128
530 IF K=83 THEN D2=D2+1/5 :: SB=SB+1/
5 :: IF SB>127 THEN SB=127
540 IF K=68 THEN D2=D2-1/5 :: SB=SB-1/
5 :: IF SB<-128 THEN SB=-128
550 DIS=DIS-ILE*15 :: D=11-INT(DIS/1000
):: IF DIS<200 THEN GOSUB 1260 ::
GOTO 570
560 IF D<9 THEN GOSUB 1220 ELSE ON D-8
GOSUB 1230,1240,1250
570 D1=D1+DEV*(D/11):: D2=D2+DEV*(D/11
)
580 IF D1>127 THEN D1=127
590 IF D1<-128 THEN D1=-128
600 IF D2>127 THEN D2=127
610 IF D2<-128 THEN D2=-128
620 CALL MOTION(#1,D1,D2)
630 IF S=0 THEN 650
640 FOR SM=2 TO 15 :: CALL MOTION(#SM,
SA,SB):: NEXT SM
650 CALL SOUND(-100,300,15)
660 TIME=TIME+1 :: IF TIME=1000 THEN 9
70
670 DISPLAY AT(1,3):"SCORE:";SC;"TIME:
";TIME :: GOTO 470
680 CALL CHARSET
690 DISPLAY AT(24,2):CHR$(92);"
";CHR$(93)
700 DISPLAY AT(23,3):CHR$(92);"
";CHR$(93)
710 DISPLAY AT(22,4):CHR$(92);"
";CHR$(93)
720 DISPLAY AT(21,5):CHR$(92);"
";CHR$(93)
730 DISPLAY AT(20,6):"
";CHR$(91)
740 DISPLAY AT(19,7):"
";CHR$(91)
750 DISPLAY AT(18,8):"
";CHR$(91)
760 DISPLAY AT(17,9):"
";CHR$(91)
770 DISPLAY AT(16,10)SIZE(1):"
SPLAY AT(16,19)SIZE(1):"
780 DISPLAY AT(15,11)SIZE(8):"
790 DISPLAY AT(14,12)SIZE(6):"
800 DISPLAY AT(13,13)SIZE(1):"
SPLAY AT(15,16)SIZE(1):"
```

```
810 DISPLAY AT(12,14)SIZE(2):"XY"
820 CALL HCHAR(11,16,32,2)
830 RETURN
840 IF SP1=0 THEN D1=INT(L-(RND*L*2)):
    :D2=INT(L-(RND*L*2)):: CALL SPRIT
    E(#1,196,7,INT(RND*256)+1,INT(RND*
    256)+1,D1/(11/D),D2/(11/D))
850 SP1=1 :: DIS=11000
860 L=L+1 :: RETURN
870 CALL COLOR(8,7,1):: CALL COLOR(8,1
    ,1)
880 CALL COINC(#1,87,124,D,C1)
890 CALL SOUND(20,888,2,999,2,10000,30
    ,-4,2)
900 IF C1=-1 THEN SP1=0 :: CALL MAGNIF
    Y(1):: CALL DELSPRITE(#1):: GOTO 9
    20
910 RETURN
920 SC=SC+1 :: FOR CS=1 TO 5 :: CALL S
    CREEN(7):: CALL SCREEN(2):: NEXT C
    S
930 CALL SOUND(500,110,2,-4,2):: CALL
    HCHAR(12,16,124,2):: CALL HCHAR(11
    ,16,124,2):: CALL SOUND(1000,110,2
    ,220,2,330,2,-8,2)
940 CALL SOUND(1,40000,30):: GOSUB 810
950 SA=0 :: SB=0 :: D=1 :: DIS=11000 ::
    L=L+2 :: GOSUB 840
960 RETURN
970 CALL CLEAR :: CALL SOUND(1000,440,
    2,550,2,660,2):: CALL SOUND(2000,7
    70,2,880,2,990,2)
980 CALL DELSPRITE(ALL)
990 SCO=SCO+SC
1000 IF SCO>=25 AND SCO-SC1>=-25 THEN TI
    ME=1000 :: SC1=SCO :: DISPLAY AT(2
    ,3):"BONUS GAME" :: GOTO 210
1010 CALL CHARSET :: CALL SCREEN(6):: C
    ALL DELSPRITE(#1,#2,#3)
1020 IF SCO>=40 THEN 1070
1030 IF SCO>=30 THEN 1090
1040 IF SCO>=20 THEN 1110
1050 IF SCO>=10 THEN 1130
1060 IF SCO>=5 THEN 1150 ELSE 1170
1070 DISPLAY AT(4,1):"A VERY GOOD BATTL
    E" :: "YOUR NAME WILL GO DOWN IN" :"H
    ISTORY AS ONE OF THE"
1080 DISPLAY AT(7,1):"GREATEST STARSHIP
     CAPTAINS" :"OF YOUR TIME." :"YOUR S
    CORE=";SC :: GOTO 1190
1090 DISPLAY AT(4,1):"YOU ARE TO BE CON
    GRADULATED" :"ON YOUR FINE MISSION.
    FEW" :"PILOTS HAVE ATTAINED SUCH"
1100 DISPLAY AT(7,1):"SUCCESS.GOOD LUCK
     ON FUTURE" :"MISSIONS." :" YOUR SCO
    RE=";SC :: GOTO 1190
1110 DISPLAY AT(4,1):"A FAIR SHOWING. T
    HE ALIENS" :"HAVE BEEN TURNED BACK
    AND" :"YOUR HOME WORLD IS SAFE."
1120 DISPLAY AT(7,1):"YOUR SCORE=";SC ::
     GOTO 1190
1130 DISPLAY AT(4,1):"YOUR FLEET WAS BA
    DLY DAMAGED" :"IN THE FIGHT, BUT YO
    U" :"MANAGED TO FOGHT OFF THE"
1140 DISPLAY AT(7,1):"ALIEN ATTACK. BET
    TER LUCK" :"NEXT TIME." :"YOUR SCORE
    =";SC :: GOTO 1190
1150 DISPLAY AT(4,1):"YOUR FLEET HAS BE
    EN" :"DESTROYED. YOU ARE THE" :"ONL
    Y SURVIVOR."
1160 DISPLAY AT(7,1):"YOUR HOME PLANET
    IS SAFE" :"AT LEAST UNTIL THE NEXT
     :"ATTACK." :"YOUR SCORE=";SC :: GOT
    O 1190
1170 DISPLAY AT(4,1):"ALL HOPE IS LOST
    IN TRYING" :"TO SAVE YOUR PLANET." :
    "YOUR MISSION HAS FAILED."
1180 DISPLAY AT(7,1):"AND YOU ARE DISGR
    ACED." :"YOUR SCORE=";SC :: GOTO 11
    90
1189 DISPLAY AT(10,1):"DO YOU WISH TO P
    LAY AGAIN" :"ENTER (Y/N)."::
```

```
1200 ACCEPT AT(11,14)SIZE(1)VALIDATE("Y
    N"):ANS$
1210 IF ANS$="N" THEN END ELSE SC,SC0,S
    C1,TIME=0 :: CALL MAGNIFY(1):: GOT
    O 190
1220 CALL MAGNIFY(1):: CALL PATTERN(#1,
    103+D):: RETURN
1230 CALL PATTERN(#1,112):: CALL MAGNIF
    Y(3):: RETURN
1240 CALL PATTERN(#1,116):: RETURN
1250 CALL PATTERN(#1,120):: RETURN
1260 CALL MAGNIFY(4):: CALL POSITION(#1
    ,P01,P02):: IF P01>0 AND P02>0 THE
    N CALL LOCATE(#1,P01-8,P02-8)
1270 IF P01<110 AND P01>56 AND P02<148
    AND P02>88 THEN GOTO 1350
1280 IF D1<-19.6 THEN R1=-19.6 :: GOTO
     1300
1290 IF D1>19.6 THEN R1=19.6 ELSE R1=D1
1300 IF D2<-19.6 THEN R2=-19.6 :: GOTO
     1320
1310 IF D2>19.6 THEN R2=19.6
1320 CALL MOTION(#1,R1*5+40*SIN(R1),R2*
    5+40*SIN(R2)):: FOR TD=1 TO 20 ::
    NEXT TD
1330 CALL DELSPRITE(#1):: SP1=0 :: DIS=
    11000 :: D=1 :: FOR MO=1 TO 15 ::
    CALL MOTION(#MO,0,0):: NEXT MO
1340 SA=0 :: SB=0 :: SP1=0 :: CALL MAGN
    IFY(1):: GOSUB 840 :: RETURN
1350 CALL POSITION(#1,D3,D4):: CALL VCH
    AR(D3/8+3,D4/8+2,54,21-D3/8):: CRA
    SH=CRASH+1 :: CALL SOUND(1000,110,
    2,220,2,10000,30,-4,2)
1360 CALL VCHAR(D3/8+3,D4/8+2,32,21-D3/
    8):: GOSUB 1280
1370 CALL SOUND(500,110,2,220,2,20000,3
    0,-8,2)
1380 CALL SOUND(500,440,2,660,2,3000,30
    ,-4,2)
1390 CALL SOUND(600,110,2,220,2,5000,30
    ,-8,2)
1400 CALL SOUND(1000,220,2,230,2,1000,3
    0,-8,2):: SA,SB=0 :: GOTO 970
1410 Z1$="0101163C424121"
1420 CALL COLOR(13,16,1)
1430 Z1$="0000000100000000"
1440 ST=2
1450 CALL CHAR(128,Z1$)
1460 CALL CHAR(129,"00"):: CALL CHAR(13
    0,"00"):: CALL CHAR(131,"00")
1470 FOR ST=2 TO 15
1480 STA1=INT(RND*256)+1 :: STA2=INT(RN
    D*256)+1
1490 CALL SPRITE(#ST,128,16,STA1,STA2)
1500 NEXT ST
1510 RETURN
1520 END
1530 DISPLAY AT(11,8):"**********"
1540 DISPLAY AT(12,8):"* FORCE 1 *"
1550 DISPLAY AT(13,8):"**********"
1560 GOSUB 590 :: DISPLAY AT(24,1):"PRE
    SS ANY KEY TO CONTINUE"
1570 CALL KEY(0,K,S):: IF S<>0 THEN CAL
    L SOUND(-1,40000,30):: CALL CLEAR
    :: RETURN ELSE CALL MAGNIFY(1)
1580 T1=INT(RND*192)+1 :: T2=INT(RND*25
    6)+1 :: CALL SPRITE(#1,196,2,T1,T2
    ,INT(RND*7)-3,INT(RND*7)-3)
1590 D=0
1600 D=D+1 :: IF D<9 THEN GOSUB 1220 EL
    SE ON D-8 GOSUB 1230,1240,1250
1610 CALL SOUND(-4000,600,(11-D)*3,400,
    (D-11)*3)
1620 CALL KEY(0,K,S):: IF S<>0 THEN CAL
    L DELSPRITE(#1):: CALL SOUND(-1,40
    000,30):: CALL CLEAR :: RETURN
1630 IF D<11 THEN 1600 ELSE CALL DELSPR
    ITE(#1):: GOTO 1570
```

DODGE 'EM

EXTENDED BASIC



Remember going to the amusement park and riding the bumper cars or *Dodge'ems*? Some people like to drive and try not to hit any other cars. Other drivers see how many cars they can hit. This computer version of *Dodge'em* has several cars randomly moving up and down the screen. The object of the game is to drive as quickly as you can from the right to the left of the screen. See what your minimum time is for crossing. A short victory melody will be played if you cross successfully (no crashing). Of course, some of you players may tire of that and try to see how many crashes you can have in each crossing or within a certain time limit.

Programming

My goal for this game was to make a game in Extended BASIC with as short a listing as possible so even the non-typers would not take too long to key in a program to run. This program is a total of 73 statements yet contains 27 moving sprites. The actual game logic is contained in 21 lines (Lines 160 to 360). You could really have fewer lines by stacking statements if you don't mind long lines.

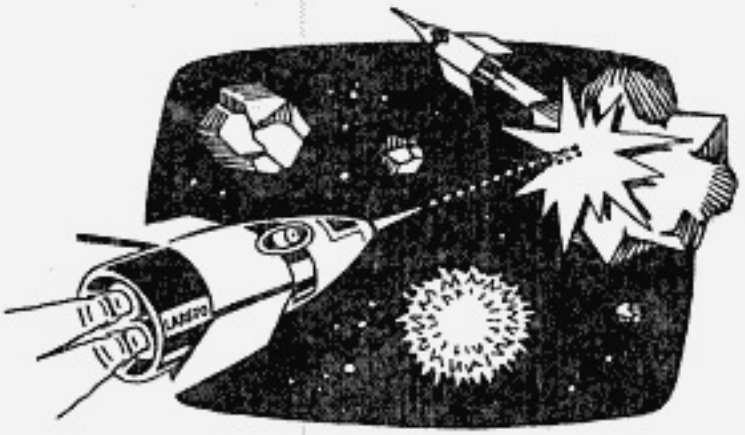
EXPLANATION OF THE PROGRAM
DODGE'EM

Line Nos.	
100-150	Introductory REMarks; branches to title screen.
160	Clears screen.
170-180	Draws left and right borders; prints "TIME:"
190-220	Places 26 cars moving vertically at random speeds.
230-240	Initializes variables; randomly places red car at right side of screen; beeps.
250-300	Depending on key pressed, sets row velocity and column velocity and moves red car.
310	Increments and prints time counter.
320	Checks coincidence for a crash.
330	If there is a crash, sounds a crashing noise and increments number of crashes.
340	If the car is not at the left border, program branches to Line 240.
350-360	Stops the red car and prints the number of crashes.
370-410	If there were no crashes, plays victory melody.
420-460	Asks if player wants to try again and branches appropriately.
470-570	Prints title screen while sounding crashing noises and defining special characters.
580-710	Prints instructions.

```
100 REM ********************
110 REM *  DODGE 'EM  *
120 REM ********************
130 REM
140 REM
150 GOTO 470
160 CALL CLEAR
170 CALL VCHAR(1,2,112,24)::: CALL VCHA
    R(1,31,112,24)
180 DISPLAY AT(1,1):"TIME:"
190 RANDOMIZE
200 FOR I=2 TO 27
210 CALL SPRITE(#I,96,5,9,1+8*I,I*(-1)^
    I+INT(RND*12+1),8)
220 NEXT I
230 RV=0 ::: CV=0 ::: T=0 ::: CR=0
240 CALL SPRITE(#1,104,7,INT(RND*180+1
    ),255)::: CALL SOUND(150,1597,0)
250 CALL KEY(0,K,S)
260 IF K=69 THEN RV=-5 ::: CV=0
270 IF K=88 THEN RV=5 ::: CV=0
280 IF K=83 THEN CV=-6 ::: RV=0
290 IF K=68 THEN CV=0 ::: RV=0
300 CALL MOTION(#1,RV,CV)
310 T=T+1 ::: DISPLAY AT(1,7):T
320 CALL COINC(ALL,C)::: IF C=0 THEN 34
    0
330 CALL SOUND(150,-6,0)::: CR=CR+1
340 CALL POSITION(#1,RO,CO)::: IF CO>16
     THEN 250
350 CALL MOTION(#1,0,0)
360 DISPLAY AT(5,1):"CRASHES: ";CR
370 IF CR>0 THEN 420
380 RESTORE 400
390 FOR I=1 TO 19 ::: READ M ::: CALL SO
    UND(100,M,1)::: NEXT I
400 DATA 262,330,392,523,392,523,330,3
    92,523,659
410 DATA 523,659,392,523,659,784,659,7
    84,784
420 DISPLAY AT(24,1):"TRY AGAIN? (Y/N)
    "
430 CALL KEY(0,KE,S)
440 IF KE=89 THEN DISPLAY AT(3,1):" "
    :: GOTO 250
450 IF KE=78 THEN CALL CLEAR ELSE 450
460 STOP
470 CALL CLEAR ::: CALL SOUND(500,-6,1,
    440,5)
480 DISPLAY AT(11,7):"D O D G E ' E M"
490 CALL SOUND(500,-7,1)
500 CALL CHAR(96,"387C7C3838387C7C38")
510 CALL SOUND(500,-5,1)
520 CALL CHAR(104,"0066FFFFFF663C18")
530 CALL COLOR(11,11,11)
540 CALL CHAR(74,"1038549219181818")
550 CALL CHAR(81,"0808080849241C08")
560 CALL CHAR(88,"1020487F4820100")
570 CALL CHAR(90,"0810402FF0204810")
580 CALL CLEAR
590 DISPLAY AT(2,7):"D O D G E ' E M"
600 DISPLAY AT(5,2):"DRIVE THE RED CAR
    FROM THE"
610 DISPLAY AT(6,2):"RIGHT SIDE OF THE
    SCREEN"
620 DISPLAY AT(7,2):"TO THE LEFT SIDE
    BY USING"
630 DISPLAY AT(8,2):"THE ARROW KEYS (
    Q X Z)."
640 DISPLAY AT(10,2):BEEP;"TRY NOT TO C
    RASH INTO"
650 DISPLAY AT(11,2):"OTHER CARS."
660 DISPLAY AT(13,2):"YOU MAY SLOW DOW
    N OR STOP"
670 DISPLAY AT(14,2):"BY PRESSING 2 (U
    T YOU MAY"
680 DISPLAY AT(15,2):"NOT BACK UP)."
690 DISPLAY AT(24,2):"PRESS ANY KEY TO
     START."
700 CALL KEY(0,KEY,S)::: IF S=0 THEN 70
    0 ELSE 160
710 END
```

TI BASIC

SPACE WAR



Space War is a two-player game written in TI BASIC. Each player has one rocket. The object of the game is to destroy your opponent either by missile fire, forcing him to crash with an asteroid, or by causing him to use up his allotment of fuel.

You can fire missiles in any of the eight directions selectable from each side of the split keyboard. Missiles emit a nerve gas that paralyzes any moving object on the screen until a hit is made or the missile goes out of range—i.e., off the screen. Firing a missile, however, does require an expenditure of fuel.

Each rocket starts out with 50 units of fuel. One unit is subtracted for each move, and a missile shot costs 5 units of fuel, so you must try to move efficiently and shoot accurately. If you run out of fuel, the game ends and the other player receives 2 points.

If your missile hits the enemy rocket, you score 5 points. If you crash into an asteroid, your opponent receives 3 points. And if you crash into each other, no points are awarded. If you shoot an asteroid you lose 1 point but the game does not stop.

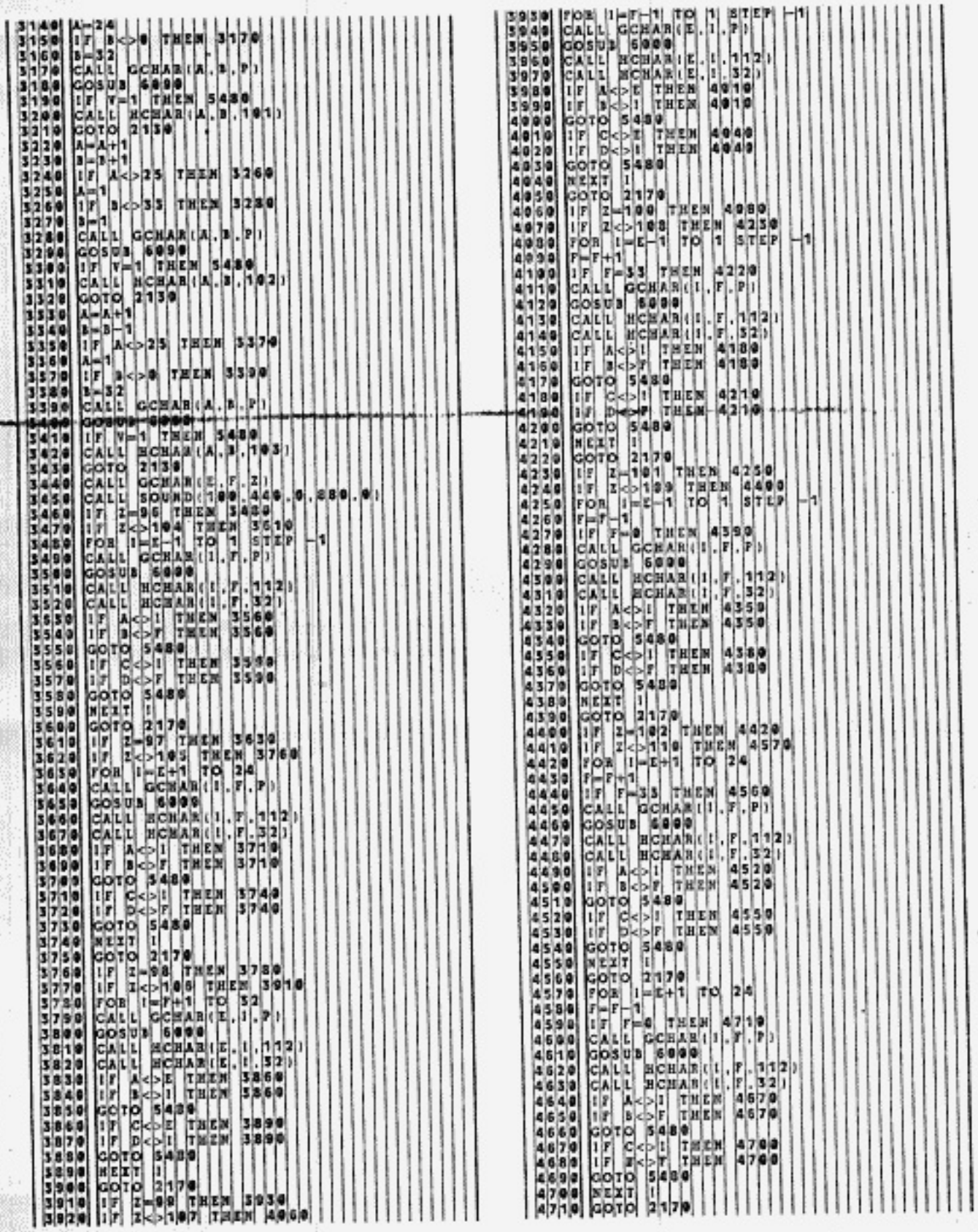
Note: If using a disk system, type CALL FILES(1) prior to RUNing. Even so, you still might encounter some conditions during play when the memory will fill and the program will halt. To eliminate this, you can delete all the instructional PRINT statements.

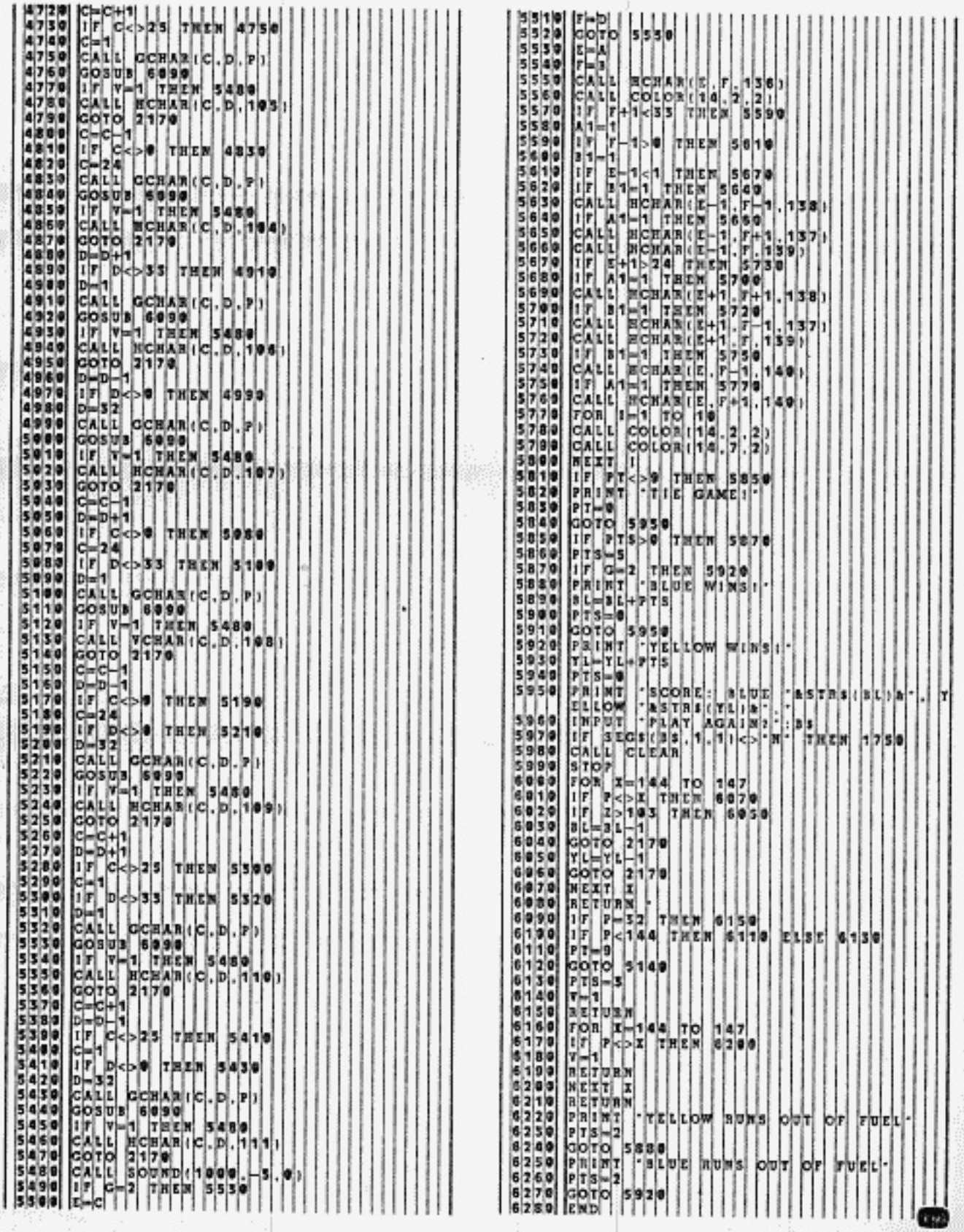
EXPLANATION OF THE PROGRAM
Space War

Line Nos.	
150-180	Clears screen, initializes fuel 50 units; makes black screen.
190-370	Definition of characters and colors for title screen and instructions. Characters 152-159 are arrows.
380-570	Draws title screen.
580-660	Draws border and blinks colors until user presses a key.
670-800	Asks if user wants instructions and waits for response.
810-1270	Prints instructions invisibly and makes white letters appear on black screen.
1280-1350	Clears screen, resets letters to white on black and defines colors for game.
1360-1790	Defines characters for graphics. Characters starting with R are the rockets in different directions; VS is for the missile; S indicates asteroids, and D crashing graphics.
1800-1880	Clears screen for game, initializes variables and draws rockets; A1, B1 are coordinates for crashing; A, B, and C, D are the rockets' coordinates.
1890-2150	Draws center asteroid, then 7 random asteroids, making sure asteroids do not overlap.
2160-2240	Receives players' input. If a key has been pressed, branches accordingly. If no key has been pressed, goes to the other player's keyboard input. Initializes variables. G indicates who is playing. V = 1 when an asteroid has been hit.
2250-2310	Procedure when yellow rocket fires a missile.
2320-2470	Procedure when yellow rocket moves.
2480-2540	Procedure when blue rocket fires missile.
2550-2700	Procedure when blue rocket moves.
2710-3460	Routines for moving the blue rocket different directions.
3470-4740	Routines for shooting missile different directions.
4750-5500	Routines for moving yellow rocket different directions.
5510-5830	Sounds crashing noise and flashes graphics.
5840-5980	Calculates and prints scores and ending remarks.
5990-6040	Prints option to play again and receives user input.
6050-6130	Subroutine to check if asteroid is shot; if so, score is decreased one point.
6140-6200	Subroutine to check if rocket is hit or if rocket hits asteroid.
6210-6260	Subroutine to check if asteroid is in that position; if so, V = 1.
6270-6330	Procedure if rocket runs out of fuel.

```
100 REM  *****************
110 REM  *  SPACE WAR  *
120 REM  *****************
130 CALL CLEAR
140 CALL SCREEN(2)
150 CALL CHAR(128,"03030303030303C3")
160 CALL CHAR(129,"C0C0C0C0C0C0C0C3")
170 CALL CHAR(120,"F0F0F0F0F0F0F0F0")
180 CALL CHAR(121,"FFFFFFFF")
190 CALL CHAR(152,"081C2A49080808FF")
200 CALL CHAR(153,"08080808492A1C08")
210 CALL CHAR(154,"0008040201FF0201")
220 CALL CHAR(155,"0010204080FF4020")
230 CALL CHAR(156,"1F0305090911204C")
240 CALL CHAR(157,"F8C0A09088804020")
250 CALL CHAR(158,"8040201108050301")
260 CALL CHAR(159,"0102040888A0C0F8")
270 A$=CHR$(157)&"  "&CHR$(152)&"  "&CHR$(156)
280 A2$=CHR$(159)&"  "&CHR$(153)&"  "&CHR$(158)
290 CALL COLOR(13,2,2)
300 CALL COLOR(15,2,2)
310 FOR I=1 TO 8
320 CALL COLOR(I,2,2)
330 NEXT I
340 PRINT TAB(4);"PRESS ANY KEY TO BEG
    IN";::::::
350 DATA 3,6,5,4,6,1,5,6,3,6,8,1,7,6,3
    ,3,11,5,11,1,3,15,2,5,15,2,5,19,
    3,7,10,5,5,12,5,3
360 DATA 5,24,1,7,23,3,13,14,5,9,21,2,
    11,21,3,2,9,5,5,5,11,2,6,1,13,5,27,1,5
370 DATA 3,10,5,5,5,12,3,3,14,5,3,9,17,5,4
    ,19,3,4,23,5,5,9,14,4,11,16,2,9,18,4,2
380 DATA 9,20,5,5,9,23,5,10,25,4,10,27,2
390 FOR I=1 TO 20
400 READ A,B,C
410 CALL HCHAR(A,B,144,C)
420 NEXT I
430 FOR I=1 TO 15
440 READ A,B,C
450 CALL VCHAR(A,B,144,C)
460 NEXT I
470 CALL HCHAR(12,26,128)
480 CALL HCHAR(12,27,129)
490 CALL COLOR(15,16,16)
500 CALL COLOR(13,16,2)
510 FOR I=1 TO 8
520 CALL COLOR(I,16,2)
530 NEXT I
540 CALL HCHAR(1,2,120,29)
550 CALL HCHAR(24,2,121,29)
560 CALL VCHAR(1,30,121,24)
570 CALL VCHAR(1,2,121,24)
580 CALL COLOR(12,7,2)
590 CALL KEY(0,T,S)
600 IF S=0 THEN 630
610 CALL COLOR(12,2,7)
620 GOTO 580
630 CALL CLEAR
640 FOR I=1 TO 8
650 CALL COLOR(I,2,2)
660 NEXT I
670 PRINT "INSTRUCTIONS?(Y OR N)";::::::
    ::::
680 FOR I=1 TO 8
690 CALL COLOR(I,16,2)
700 NEXT I
710 CALL KEY(0,T,S)
720 IF S=0 THEN 710
730 FOR I=1 TO 8
740 CALL COLOR(I,2,2)
750 NEXT I
760 IF T=78 THEN 1230
770 PRINT TAB(8);"SPACE WAR"
780 PRINT "A GAME FOR TWO PLAYERS"
790 PRINT :"THE OBJECT OF THE GAME IS
    TO DESTROY THE OPPONENT BY"
800 PRINT "MISSLE FIRING OR BY FORC
    ING"
810 PRINT "HIM INTO AN ASTEROID.":::
820 PRINT TAB(11);"KEYS":::
830 PRINT "TO MOVE YOUR ROCKET. PRESS"
840 PRINT "KEYS AS SHOWN BELOW:":::
850 PRINT "PLAYER 1","PLAYER 2"
860 PRINT TAB(2);"A","TAB(16);"A"
870 PRINT TAB(2);"W","E","S";TAB(17);"W
    E","S"
880 PRINT TAB(2);"D";TAB(15);"D"
890 PRINT TAB(2);"X";TAB(16);"X"
900 PRINT TAB(2);"Z";TAB(17);"Z"
910 PRINT TAB(2);"A";TAB(16);"A"
920 PRINT "PLAYER 1 USES Q TO F
    IRE."
930 PRINT "AND PLAYER 2 USES P."
940 PRINT "PRESS ANY KEY TO CONTINUE"
950 FOR I=1 TO 8
960 CALL COLOR(I,16,2)
970 NEXT I
980 CALL COLOR(16,16,2)
990 CALL KEY(0,T,S)
1000 IF S=0 THEN 990
1010 CALL COLOR(16,2,2)
1020 FOR I=1 TO 8
1030 CALL COLOR(I,2,2)
1040 NEXT I
1050 PRINT TAB(9);"ASTEROIDS":::
1060 PRINT "THIS WAR TAKES PLACE IN AN"
1070 PRINT "ASTEROID BELT. YOU CANNOT"
1080 PRINT "SHOOT THROUGH THEM, AND"
1090 PRINT "RUNNING INTO ONE IS FATAL."
1100 PRINT :::TAB(9);"MISSLES":::
1110 PRINT "MISSLES CAN BE FIRED AT"
1120 PRINT "ANY ANGLE. THEY EMIT A GAS"
1130 PRINT "THAT PARALYZES ANY MOVING"
1140 PRINT "OBJECTS ON THE SCREEN FROM"
1150 PRINT "THE TIME THAT IT WAS FIRED"
1160 PRINT "UNTIL IT HITS OR DISAPPEARS
    ":::::
1170 PRINT "PRESS ANY KEY TO CONTINUE"
1180 FOR I=1 TO 8
1190 CALL COLOR(I,16,2)
1200 NEXT I
1210 CALL KEY(0,T,S)
1220 IF S=0 THEN 1210
1230 CALL CLEAR
1240 FOR I=1 TO 8
1250 CALL COLOR(I,16,2)
1260 NEXT I
1270 CALL COLOR(9,5,2)
1280 CALL COLOR(10,11,2)
1290 CALL COLOR(11,16,2)
1300 CALL COLOR(14,7,2)
1310 RU$="080808081C1C3E2A22"
1320 RD$="222A3E1C1C080808"
1330 RR$="0000001038783838"
1340 RL$="0000001C7E1C0814"
1350 RUR$="0102144789C28881"
1360 RUL$="8040281E391412A4"
1370 RDR$="1088289C7814A281"
1380 RDL$="0810145391E28488"
1390 B$="0000001818"
1400 S1$="7F7F373731F0F03"
1410 S2$="FEFEFCFCF8F0C"
1420 S3$="0080F0F8FCFCFEFE"
1430 S4$="00C0F0F8FCFEFEFE"
1440 D$="0018243C7EFFFF"
1450 D1$="0102040810204080"
1460 D2$="8040201008040201"
1470 D3$="1818181818181818"
1480 D4$="0000007EFFFF"
1490 CALL CHAR(96,RU$)
1500 CALL CHAR(97,RD$)
1510 CALL CHAR(98,RR$)
1520 CALL CHAR(99,RL$)
1530 CALL CHAR(100,RUR$)
1540 CALL CHAR(101,RUL$)
1550 CALL CHAR(102,RDR$)
```

```
1560 CALL CHAR(103,RDLS$)
1570 CALL CHAR(104,RUS$)
1580 CALL CHAR(105,RDS$)
1590 CALL CHAR(106,RRS$)
1600 CALL CHAR(107,RLS$)
1610 CALL CHAR(108,RUDS$)
1620 CALL CHAR(109,RDLS$)
1630 CALL CHAR(110,RDRS$)
1640 CALL CHAR(111,RDLS$)
1650 CALL CHAR(112,BS$)
1660 CALL CHAR(136,DD1S$)
1670 CALL CHAR(137,DD1S$)
1680 CALL CHAR(138,DD2S$)
1690 CALL CHAR(139,DD3S$)
1700 CALL CHAR(140,DD4S$)
1710 CALL CHAR(144,SS1$)
1720 CALL CHAR(145,SS2$)
1730 CALL CHAR(146,SS3$)
1740 CALL CHAR(147,SS4$)
1750 CALL CLEAR
1760 BPR=-50
1770 YPR=-50
1780 A1=0
1790 B1=0
1800 A=12
1810 B=3
1820 C=12
1830 D=30
1840 CALL HCHAR(A,B,98)
1850 CALL HCHAR(C,D,107)
1860 CALL COLOR(15,16,2)
1870 CALL HCHAR(15,16,144)
1880 CALL HCHAR(15,17,145)
1890 CALL HCHAR(12,16,146)
1900 CALL HCHAR(12,17,147)
1910 FOR I=1 TO 7
1920 RANDOMIZE
1930 SH=INT(20*RND)+3
1940 SV=INT(23*RND)+5
1950 FOR I1=0 TO 1
1960 CALL GCHAR(SH,SV+I1,P)
1970 GOSUB 6160
1980 IF V=1 THEN 2060
1990 NEXT I1
2000 FOR I1=0 TO 1
2010 CALL GCHAR(SH-1,SV+I1,P)
2020 GOSUB 6160
2030 IF V=1 THEN 2060
2040 NEXT I1
2050 GOTO 2080
2060 V=0
2070 GOTO 1920
2080 CALL HCHAR(SH,SV,144)
2090 CALL HCHAR(SH,SV+1,145)
2100 CALL HCHAR(SH-1,SV,146)
2110 CALL HCHAR(SH-1,SV+1,147)
2120 NEXT I
2130 CALL KEY(2,K1,S1)
2140 G=0
2150 V=0
2160 IF S1<>0 THEN 2220
2170 CALL KEY(1,K,S)
2180 G=0
2190 V=0
2200 IF S<>0 THEN 2450
2210 GOTO 2130
2220 IF K1<>11 THEN 2290
2230 YPR=YPR-5
2240 IF YPR<0 THEN 6220
2250 CALL GCHAR(C,D,P)
2260 CALL HCHAR(C,D,32)
2270 CALL SOUND(200,-6,0,440,0)
2280 G=1
2290 GOTO 3440
2300 YPR=YPR-1
2310 IF YPR<0 THEN 6220
2320 CALL GCHAR(C,D,P)
2330 CALL HCHAR(C,D,32)
2340 CALL SOUND(200,-6,0,440,0)
2350 IF K1+1=1 THEN 4720
2360 IF K1+1=5 THEN 4880
2370 IF K1+1=3 THEN 4880
2380 IF K1+1=7 THEN 4960
2390 IF K1+1=6 THEN 5040
2400 IF K1+1=4 THEN 5150
2410 IF K1+1=14 THEN 5250
2420 IF K1+1=15 THEN 5370
2430 CALL HCHAR(C,D,P)
2440 GOTO 2170
2450 IF K<>18 THEN 2520
2460 BPR=BPR-5
2470 IF BPR<0 THEN 6250
2480 L=A
2490 F=B
2500 G=1
2510 GOTO 3440
2520 BPR=BPR-1
2530 IF BPR<0 THEN 6250
2540 CALL GCHAR(A,B,P)
2550 CALL HCHAR(A,B,32)
2560 CALL SOUND(200,-6,0,440,0)
2570 G=2
2580 IF K=5 THEN 2680
2590 IF K+1=1 THEN 2760
2600 IF K=3 THEN 2840
2610 IF K=2 THEN 2920
2620 IF K=6 THEN 3000
2630 IF K=4 THEN 3110
2640 IF K=14 THEN 3220
2650 IF K=15 THEN 3330
2660 CALL HCHAR(A,B,P)
2670 GOTO 2130
2680 A=A-1
2690 IF A<>0 THEN 2710
2700 A=24
2710 CALL GCHAR(A,B,P)
2720 GOSUB 6090
2730 IF V=1 THEN 5480
2740 CALL HCHAR(A,B,96)
2750 GOTO 2130
2760 A=A+1
2770 IF A<>25 THEN 2790
2780 A=1
2790 CALL GCHAR(A,B,P)
2800 GOSUB 6090
2810 IF V=1 THEN 5480
2820 CALL HCHAR(A,B,97)
2830 GOTO 2130
2840 B=B+1
2850 IF B<>33 THEN 2870
2860 B=1
2870 CALL GCHAR(A,B,P)
2880 GOSUB 6090
2890 IF V=1 THEN 5480
2900 CALL HCHAR(A,B,98)
2910 GOTO 2130
2920 B=B-1
2930 IF B<>0 THEN 2950
2940 B=32
2950 CALL GCHAR(A,B,P)
2960 GOSUB 6090
2970 IF V=1 THEN 5480
2980 CALL HCHAR(A,B,99)
2990 GOTO 2130
3000 A=A-1
3010 B=B+1
3020 IF A<>0 THEN 3040
3030 A=24
3040 IF B<>33 THEN 3060
3050 B=1
3060 CALL GCHAR(A,B,P)
3070 GOSUB 6090
3080 IF V=1 THEN 5480
3090 CALL HCHAR(A,B,100)
3100 GOTO 2130
3110 A=A-1
3120 B=B-1
3130 IF A<>0 THEN 3150
```





TI BASIC

MAZE RACE

Maze Race is a game written in TI BASIC for two players; one controls the red soldier, and one controls the blue soldier. The game starts out with the opposing soldiers lost at the ends of a forest maze. The object is to reach the safe zone across the field without meeting the enemy. The first soldier to cross his boundary into safety (through the entrance) wins the round, and the game continues until one soldier scores ten times. If the soldiers collide, neither one scores.

The maze is drawn randomly by the computer, so if an impossible maze is drawn (an entrance blocked or a soldier surrounded), it may be redrawn by answering the "Change Maze?" option with "Y" for yes.

The red soldier is moved by pressing the arrow keys on the left keyboard. The blue soldier is moved by pressing I for up, J for left, K for right, and M for down. You may wish to use the Video Games I Command Cartridge overlay. No diagonal moves are allowed, and a soldier cannot go through a barrier. Once a key is pressed, the soldier moves in that direction until another key is pressed.

The difficulty of the maze may be altered by adjusting the PRINT statements 220-560. The & is a blank space on the maze, and # is a barrier.

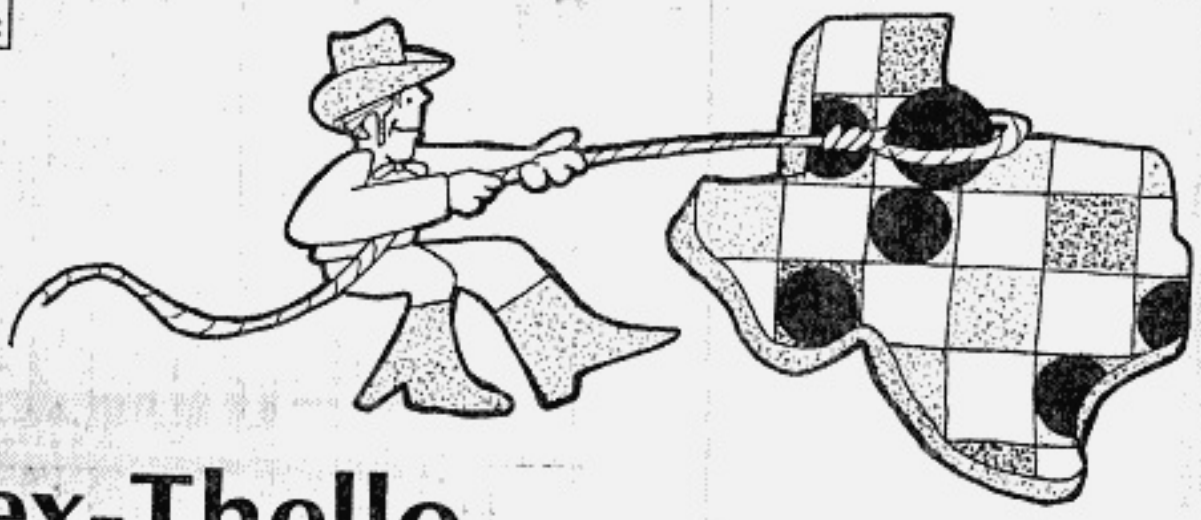
EXPLANATION OF THE PROGRAM
Maze Race

170	Branches to title screen and instructions.
180-210	Subroutine to print messages on screen below maze.
220-570	Subroutines to print maze a line at a time.
580-700	Clears screen and prints maze. Lines of maze are chosen randomly then printed.
710-740	Places soldiers at opposite ends of maze in random horizontal position.
750-810	Prints message, "CHANGE MAZE?", waits for response and branches accordingly.
820-900	Initializes variables. RX, RY, BX, and BY are directional increments. RXC, RYC, BXC, and BYC are coordinates for the red and blue soldiers. RED and BLUE = 1 for a win, 0 for a loss. Sounds a "beep" to start game.
910-1070	Reads red soldier's keyboard entry to move.
1080-1230	Checks where soldier will move and redraws soldier. Checks location for space, block, enemy entrance, or his goal.
1240-1400	Reads blue soldier's keyboard entry to move.
1410-1580	Checks blue soldier's move and location.
1590-1690	Routine if soldiers collide.
1700-1760	Prints message when one soldier wins.
1770-1940	Prints scores.
1950-2000	Asks "TRY AGAIN?" and branches accordingly.
2010-2180	Prints title screen and defines characters and colors; asks if instructions are needed and waits for response.
2190-2270	Prints instructions if desired.

```
100 REM *************
110 REM * MAZE RACE *
120 REM *************
130 REM
140 REM
150 REM
160 REM
170 GOTO 2010
180 FOR J=1 TO LEN(M$)
190 CALL HCHAR(X,J,ASC(SEG$(M$,J,1)))
200 NEXT J
210 RETURN
220 PRINT "&##&#&#&&&&&&&#&&#&&&&&##&#&"
230 RETURN
240 PRINT "&####&&&#&&&&##&&#&&#&&#&&&&"
250 RETURN
260 PRINT "&&&&&&&#&#&#&&#&&&&#&&&&#&&#"
270 RETURN
280 PRINT "&#&&&#&&&&&&&&&#&&&&&#&&&&&#"
290 RETURN
300 PRINT "&&&#&&##&&&#&&#&&#&&&&#&&#&&"
310 RETURN
320 PRINT "&&##&&##&&##&&&&&#&&&&&&#&&&"
330 RETURN
340 PRINT "#&&#&&##&#&&&&&&&&##&&&###&&"
350 RETURN
360 PRINT "&#&&&#&&&&#&&###&##&&&#&&&&#"
370 RETURN
380 PRINT "&&#&&&&&#&&&&&&&#&&&#&&&&&#&"
390 RETURN
400 PRINT "&&##&&&&##&&&&&&##&&&#&&&&&&"
410 RETURN
420 PRINT "#&#&&&&##&&&&##&&&&&&#&&#&&#"
430 RETURN
440 PRINT "#&&&#&&&&#&&&&#&&&&&&#&&&#&&"
450 RETURN
460 PRINT "&#&&&#&&&&&##&&&&&&&&#&&&&##"
470 RETURN
480 PRINT "&&#&&&&#&&&&&&&#&&&#&&&&#&&&"
490 RETURN
500 PRINT "#&&&#&&&#&#&#&&&&&&&#&&&&##&"
510 RETURN
520 PRINT "&##&&&##&&&&&###&&#&&&#&&&&&"
530 RETURN
540 PRINT "&&&#&&&&&#&&&&&&#&&&&&&#&&&&"
550 RETURN
560 PRINT "&&&&##&&&&##&&&#&&&&##&&&&&&"
570 RETURN
580 CALL CLEAR
590 RANDOMIZE
600 CALL HCHAR(23,3,35,28)
610 FOR I=2 TO 20
620 AA=INT(18*RND)+1
630 ON AA GOSUB 220,240,260,280,300,320,340,360,380,400,420,440,460,480,500,520,540,560
640 NEXT I
650 PRINT :::
660 CALL HCHAR(21,3,35,28)
670 CALL VCHAR(1,2,113,21)
680 CALL VCHAR(1,30,105,21)
690 CALL VCHAR(4,2,38,3)
700 CALL VCHAR(16,30,38,3)
710 RXC=INT(18*RND)+2
720 CALL HCHAR(RXC,3,104)
730 BXC=INT(18*RND)+2
740 CALL HCHAR(BXC,29,112)
750 M$="CHANGE MAZE? (Y/N) "
760 X=24
770 GOSUB 180
780 CALL KEY(0,KEY,S)
790 IF KEY=89 THEN 580
800 IF KEY<>78 THEN 780
810 CALL HCHAR(24,1,32,10)
820 RX=0
830 RY=0
840 BX=0
850 BY=0
860 RYC=5
870 BYC=29
880 RED=0
890 BLUE=0
900 CALL SOUND(150,1397,2)
910 CALL KEY(1,K1,S1)
920 IF S1=0 THEN 1080
930 IF K1<>5 THEN 970
940 RX=-1
950 RY=0
960 GOTO 1080
970 IF K1<>3 THEN 1010
980 RY=1
990 RX=0
1000 GOTO 1080
1010 IF K1+1<>1 THEN 1050
1020 RX=1
1030 RY=0
1040 GOTO 1080
1050 IF K1<>2 THEN 1080
1060 RY=-1
1070 RX=0
1080 IF RYC+RY>1 THEN 1120
1090 M$="RED WENT WRONG WAY-BLUE WON."
1100 BLUE=1
1110 GOTO 1440
1120 CALL GCHAR(RXC+RX,RYC+RY,CC)
1130 IF CC=38 THEN 1170
1140 IF CC=112 THEN 1590
1150 CALL SOUND(150,-5,0)
1160 GOTO 1240
1170 CALL HCHAR(RXC,RYC,38)
1180 RXC=RXC+RX
1190 RYC=RYC+RY
1200 CALL HCHAR(RXC,RYC,104)
1210 IF RYC<>30 THEN 1240
1220 RED=1
1230 GOTO 1700
1240 CALL KEY(2,K2,S2)
1250 IF S2=0 THEN 1410
1260 IF K2<>5 THEN 1300
1270 BX=-1
1280 BY=0
1290 GOTO 1410
1300 IF K2<>3 THEN 1340
1310 BY=1
1320 BX=0
1330 GOTO 1410
1340 IF K2+1<>1 THEN 1380
1350 BX=1
1360 BY=0
1370 GOTO 1410
1380 IF K2<>2 THEN 1410
1390 BY=-1
1400 BX=0
1410 IF BYC+BY<31 THEN 1470
1420 M$="BLUE WENT WRONG WAY-RED WON."
1430 RED=1
1440 X=22
1450 GOSUB 180
1460 GOTO 1770
1470 CALL GCHAR(BXC+BX,BYC+BY,DD)
```

```
1480 IF DD=38 THEN 1520
1490 IF DD=104 THEN 1590
1500 CALL SOUND(150,-7,0)
1510 GOTO 910
1520 CALL HCHAR(BXC,BYC,38)
1530 BXC=BXC+BX
1540 BYC=BYC+BY
1550 CALL HCHAR(BXC,BYC,112)
1560 IF BYC<>2 THEN 910
1570 BLUE=1
1580 GOTO 1700
1590 CALL HCHAR(BXC,BYC,38)
1600 CALL HCHAR(RXC,RYC,120,2)
1610 CALL HCHAR(RXC+RX,RYC+RY,120,2)
1620 FOR I=1 TO 15
1630 CALL COLOR(12,7,6)
1640 CALL COLOR(12,6,7)
1650 NEXT I
1660 M$="DUEL!! BOTH SOLDIERS DIE."
1670 X=23
1680 GOSUB 180
1690 GOTO 1930
1700 CALL HCHAR(23,1,32,64)
1710 IF BLUE=1 THEN 1740
1720 M$="RED WINS THIS TIME."
1730 GOTO 1750
1740 M$="BLUE WINS THIS TIME."
1750 X=22
1760 GOSUB 180
1770 M$="TOTAL SCORE:  RED       BLUE "
1780 X=23
1790 GOSUB 180
1800 REDS=REDS+RED
1810 BLUES=BLUES+BLUE
1820 IF REDS<10 THEN 1860
1830 CALL HCHAR(23,19,49)
1840 CALL HCHAR(23,20,48)
1850 GOTO 1910
1860 CALL HCHAR(23,19,REDS+48)
1870 IF BLUES<10 THEN 1910
1880 CALL HCHAR(23,29,49)
1890 CALL HCHAR(23,30,48)
1900 GOTO 2000
1910 CALL HCHAR(23,29,BLUES+48)
1920 IF REDS=10 THEN 2000
1930 FOR DELAY=1 TO 1000
1940 NEXT DELAY
1950 PRINT "TRY AGAIN? (Y/N)"
1960 CALL KEY(0,KEY,S)
1970 IF S=0 THEN 1960
1980 IF KEY=89 THEN 580
1990 IF KEY<>78 THEN 1950
2000 STOP
2010 CALL CLEAR
2020 PRINT TAB(7);"**************":TAB(7);"*            *"
2030 PRINT TAB(7);"* MAZE RACE  *":TAB(7);"*            *"
2040 PRINT TAB(7);"**************":::::::::::
2050 CALL CHAR(38,"0")
2060 CALL CHAR(35,"FFFFFFFFFFFFFFFF")
2070 CALL COLOR(1,13,1)
2080 CALL CHAR(104,"18187E1818242424")
2090 CALL CHAR(105,"FFFFFFFFFFFFFFFF")
2100 CALL COLOR(10,7,1)
2110 CALL CHAR(120,"49221449142249")
2120 CALL CHAR(112,"18187E1818242424")
2130 CALL CHAR(113,"FFFFFFFFFFFFFFFF")
2140 CALL COLOR(11,6,1)
2150 PRINT "WANT INSTRUCTIONS? (Y/N)"
2160 CALL KEY(0,KEY,S)
2170 IF KEY=78 THEN 580
2180 IF KEY<>89 THEN 2160
2190 CALL CLEAR
2200 PRINT "TWO OPPOSING SOLDIERS ARE":"LOST IN A JUNGLE MAZE."
2210 PRINT ::"EACH MAN MUST FIND HIS WAY"::"TO THE OPPOSITE BORDER"
2220 PRINT "TO BE SAFE."::"DO NOT COLLIDE OR ELSE"
2230 PRINT "BOTH SOLDIERS WILL DIE."
2240 PRINT :::"PRESS ENTER TO START."
2250 CALL KEY(0,KEY,S)
2260 IF KEY=13 THEN 580 ELSE 2250
2270 END
```

TI BASIC



Tex-Thello

Tex-Thello is a microcomputer version of the popular Othello (a trademark of Gabriel Industries, Inc.) board game. The program written in TI BASIC, pits the human player against the computer for an exciting game on three levels of difficulty: On Level 1, the computer just tries to capture the most markers. On Level 3 (the highest level), the computer takes into account the edge squares and corner squares—thus providing it with more of a theoretical advantage. Level 2 is an intermediate level. The program will check for illegal moves (sounding a warning tone within 30 seconds) and change the color of "captured" markers according to the moves.

Game Rules

1. Since the first four squares in the middle of the board must be occupied (in "checkerboard fashion") first, the program automatically provides this initial setup.

2. The player alternates turns with the computer by entering the grid coordinates for a move. A move consists of placing a color square so that it "captures" (by completing the outflanking of) one or more of the opposite color squares. The computer will then change all the captured squares to the opposite color.

3. A move must always consist of capturing at least one square.

4. If a legal move cannot be made, it then becomes the opponent's turn to move.

5. Capturing may be accomplished horizontally, vertically, or diagonally in one or more rows or directions.

6. The game is over either when the board is filled with color squares, when it is not possible for either opponent to move, or when the board is filled (or partially filled) with all one color. The opponent with the most squares is the winner.

EXPLANATION OF THE PROGRAM

Tex-Thello

Line Nos.	
160	Dimensions arrays for squares captured.
170-240	Stores the name "COMPUTER" for player.
250-400	Option screens; user presses a key for choices.
410-510	Players input names; stored in PLAY(1,10).
520-610	Initializes positions of board.
620-730	Prints labels for game.
740-920	Defines graphics characters and colors.
[illegible]	Prints starting *Tex-Thello* board on row number [illegible]
[illegible]	Initializes starting four positions.
[illegible]	Initializes squares around four center squares; [illegible]
[illegible]	Clears for first player on move number 5.
[illegible]	Prints player's name (computer) and black
[illegible]	squares indicating whose move.
1240-1330	Player presses column number then row number for move.
1340-1360	Computer prints move.
1370-1480	Checks for legal move.
1490-1550	Sets values of surrounding squares to zero.
1560-1620	Shows move on screen and switches appropriate captured squares; increments TURN (number of moves).
1630-1740	Checks to see if board still contains two colors, otherwise branches to end of game.
1750-1790	Changes player number for next turn and branches to beginning of main loop.
1800-2040	Tallies squares for each player and prints score.
2050-2100	Asks if player wants to play again; branches appropriately or ends program.
2110-2250	Subroutine to check if there is a legal move.
2260-2510	Subroutine to [illegible]
2520-2820	Subroutine to [illegible]
[illegible]	Subroutine to [illegible]
2950-4240	Subroutine to calculate computer's move. EXTRA is the number of squares that can be captured; HARD is the level of difficulty (1, 2, 3). For the different levels, the board positions have different values.

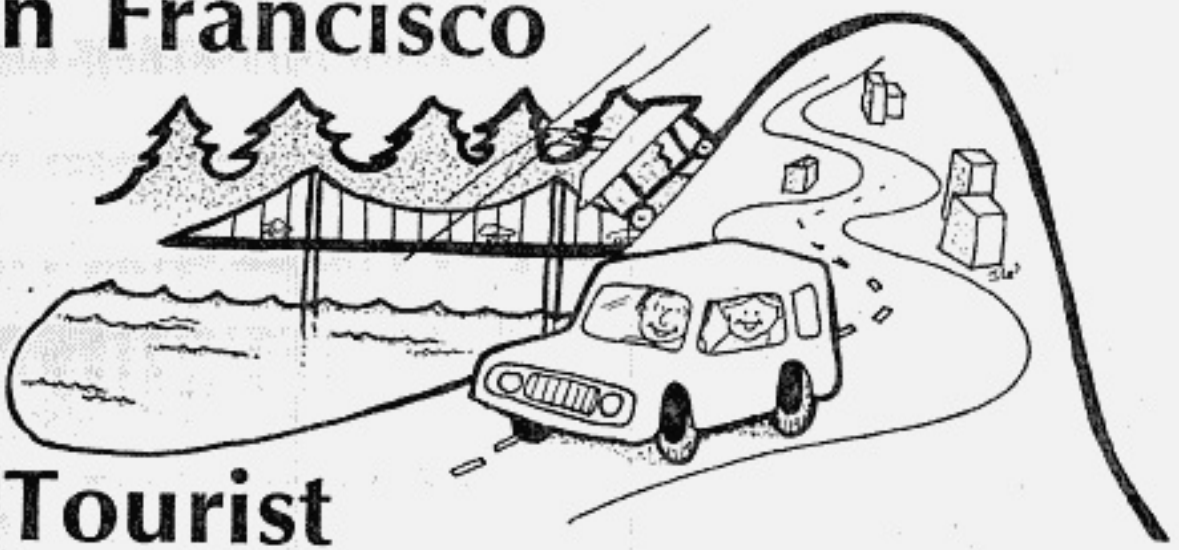
```
100 REM  *********
110 REM  * TEX-TELLO *
120 REM  *********
130 REM
140 REM
150 REM
160 DIM EX(18),EY(18)
170 DATA 67,79,77,80,85,84,69,82
180 RESTORE
190 COMPLAY=0
200 CALL CLEAR
210 FOR I=1 TO 8
220 READ PLAY(1,I)
230 PLAY(2,I)=PLAY(1,I)
240 NEXT I
250 PRINT TAB(8)::"TEX-TELLO":::::
260 PRINT "CHOOSE:::::"1  ONE PLAYER V
    S. COMPUTER":::"2  TWO PLAYERS":::::
270 CALL KEY(0,K,S)
280 IF (K<49)+(K>50)=-1 THEN 270
290 IF K=50 THEN 410
300 CALL CLEAR
310 PRINT "CHOOSE:::::"1  EASY GAME":::
    "2  INTERMEDIATE GAME":::"3  HARD G
    AME":::::
320 CALL KEY(0,K,S)
330 IF (K<49)+(K>51)=-1 THEN 320
340 HARD=K-49
350 CALL CLEAR
360 PRINT "CHOOSE -- COMPUTER PLAYS":::
    :"1  FIRST--RED":::"2  SECOND--YELL
    OW":::::
370 CALL KEY(0,K,S)
380 IF (K<49)+(K>50)=-1 THEN 370
390 COMPLAY=K-48
400 IF COMPLAY=1 THEN 470
410 PRINT :::"FIRST PLAYER NAME (RED)"
420 INPUT Z1$
430 FOR I=1 TO 10
440 CALL GCHAR(23,I+4,PLAY(1,I))
450 NEXT I
460 IF COMPLAY=2 THEN 520
470 PRINT :::"SECOND PLAYER NAME (YELLO
    W)"
480 INPUT Z1$
490 FOR I=1 TO 10
500 CALL GCHAR(23,I+4,PLAY(2,I))
510 NEXT I
520 FOR I=1 TO 3
530 DIR(I)=I-2
540 NEXT I
550 FOR I=0 TO 9
560 FOR J=0 TO 9
570 A(I,J)=-1
580 IF (I*J)*(I-9)*(J-9)<>0 THEN 606
590 A(I,J)=A(I,J)+1
600 NEXT J
610 NEXT I
620 CALL CLEAR
630 PRINT TAB(10);"X=     Y="
640 ES="FFFFFFFFFFFFFFFF"
650 CALL CHAR(120,ES)
660 CALL COLOR(12,2,16)
670 FOR I=1 TO 8
680 Y=3+2*I
690 X=Y+6
700 CD=48+I
710 CALL VCHAR(Y,X,CD)
720 CALL VCHAR(4,X,CD)
730 NEXT I
740 AS="FF8080808080808080"
750 BS="FF0101010101010101"
760 CS="8080808080808080FF"
770 DS="0101010101010101FF"
780 CALL CHAR(96,AS)
790 CALL CHAR(97,BS)
800 CALL CHAR(98,CS)
810 CALL CHAR(99,DS)
820 CALL CHAR(104,AS)
830 CALL CHAR(105,BS)
840 CALL CHAR(106,CS)
850 CALL CHAR(107,DS)
860 CALL CHAR(112,BS)
870 CALL CHAR(113,BS)
880 CALL CHAR(114,CS)
890 CALL CHAR(115,DS)
900 CALL COLOR(9,2,16)
910 CALL COLOR(10,2,9)
920 CALL COLOR(11,2,11)
930 TYPE=1
940 FOR X=1 TO 8
950 FOR Y=1 TO 8
960 GOSUB 2270
970 NEXT Y
980 NEXT X
990 FOR X=-4 TO 5
1000 FOR Y=-4 TO 5
1010 IF X=Y THEN 1050
1020 TYPE=2
1030 A(X,Y)=3
1040 GOTO 1070
1050 TYPE=3
1060 A(X,Y)=6
1070 GOSUB 2530
1080 NEXT Y
1090 NEXT X
1100 FOR I=3 TO 6
1110 FOR J=3 TO 6
1120 IF A(I,J)<>1 THEN 1140
1130 A(I,J)=8
1140 NEXT J
1150 NEXT I
1160 PL=1
1170 TURN=5
1180 REM  BEGIN MAIN LOOP
1190 FOR I=1 TO 10
1200 CALL HCHAR(23,I+7,PLAY(PL,I))
1210 NEXT I
1220 CALL VCHAR(25,25,120)
1230 CALL VCHAR(23,28,120)
1240 IF COMPLAY=PL THEN 1340
1250 CALL KEY(0,RE,ST)
1260 IF ST<>1 THEN 1250
1270 X=RE-48
1280 CALL VCHAR(23,23,RE)
1290 CALL KEY(0,RE,ST)
1300 IF ST<>1 THEN 1290
1310 Y=RE-48
1320 CALL VCHAR(23,28,RE)
1330 GOTO 1570
1340 GOSUB 2960
1350 CALL VCHAR(23,25,48+X)
1360 CALL VCHAR(23,28,48+Y)
1370 IF X<1 THEN 1460
1380 IF X>8 THEN 1460
1390 IF Y<1 THEN 1460
1400 IF Y>8 THEN 1460
1410 IF A(X,Y)>1 THEN 1460
1420 GOSUB 2530
1430 IF EXT=A>0 THEN 1490
1440 GOSUB 2120
1450 IF SW=0 THEN 1640
1460 CALL SOUND(500,200,3)
1470 IF COMPLAY=PL THEN 1640
1480 GOTO 1220
1490 A(X,Y)=PL+2
1500 FOR I=X-1 TO X+1
1510 FOR J=Y-1 TO Y+1
1520 IF A(I,J)<>1 THEN 1540
1530 A(I,J)=0
1540 NEXT J
1550 NEXT I
1560 TURN=TURN+1
1570 TYPE=2
1580 IF PL=1 THEN 1600
1590 TYPE=3
1600 GOSUB 2270
1610 GOSUB 2530
```

```
1620 GOSUB 2840
1630 IF TURN=65 THEN 1810
1640 IF PL=1 THEN 1670
1650 A1=4
1660 GOTO 1680
1670 A1=3
1680 FOR I=-1 TO 0
1690 FOR J=-1 TO 0
1700 IF A(I,J)<=1 THEN 1720
1710 IF A(I,J)<>A1 THEN 1750
1720 NEXT J
1730 NEXT I
1740 GOTO 1810
1750 IF PL=1 THEN 1780
1760 PL=1
1770 GOTO 1190
1780 PL=2
1790 GOTO 1190
1800 REM END MAIN LOOP AND BEGIN TOTAL
     S REGION
1810 TOT1=0
1820 TOT2=0
1830 FOR I=1 TO 8
1840 FOR J=1 TO 8
1850 IF A(I,J)=3 THEN 1880
1860 IF A(I,J)<>4 THEN 1900
1870 TOT2=TOT2+1
1880 GO TO 1900
1890 TOT1=TOT1+1
1900 NEXT J
1910 NEXT I
1920 IF TOT2<>TOT1 THEN 1950
1930 PRINT "DRAW 32 TO 32"
1940 GO TO 2050
1950 PRINT "AT A SCORE OF";TOT1;"TO";
     TOT2;"."
1970 PRINT "WINS."
1980 IF TOT1>TOT2 THEN 2010
1990 PW=2
2000 GO TO 2020
2010 PW=1
2020 FOR I=1 TO 10
2030 CALL HCHAR(23,I+1,PLAY(PW,I))
2040 NEXT I
2050 PRINT :"PLAY AGAIN? (Y/N)";
2060 CALL KEY(0,K,S)
2070 IF K=89 THEN 180
2080 IF K<>78 THEN 2060
2090 STOP
2100 REM
2110 REM IS THERE A LEGAL MOVE?
2120 SW=0
2130 ZZX=X
2140 ZZY=Y
2150 FOR X=1 TO 8
2160 FOR Y=1 TO 8
2170 IF A(X,Y)<>0 THEN 2210
2180 GOSUB 2550
2190 IF EXTRA=0 THEN 2210
2200 SW=1
2210 NEXT Y
2220 NEXT X
2230 X=ZZX
2240 Y=ZZY
2250 RETURN
2260 REM - COLOR ONTO BOARD ROUTINE
2270 IF TYPE<>1 THEN 2330
2280 CALL SCREEN(2)
2290 CALL HCHAR(SS1,SS2,SS3,SS4)
2300 CALL HCHAR(SS1,SS2,SS3,SS4)
2310 CALL HCHAR(SS1,SS2,SS3,SS4)
2320 GO TO 2350
2330 IF TYPE<>2 THEN 2390
2340 SS1=104
2350 SS2=105
2360 SS3=106
2370 SS4=107
2380 GO TO 2430
2390 SS1=112
2400 SS2=113
2410 SS3=114
2420 SS4=115
2430 X1=7+2*X
2440 X2=X1+1
2450 Y1=3+2*Y
2460 Y2=Y1+1
2470 CALL HCHAR(Y1,X1,SS1)
2480 CALL HCHAR(Y1,X2,SS2)
2490 CALL HCHAR(Y2,X1,SS3)
2500 CALL HCHAR(Y2,X2,SS4)
2510 RETURN
2520 REM EXTRA SQUARES
2530 EXTRA=1
2540 FOR I=-1 TO 3
2550 FOR J=-1 TO 3
2560 U=X+DIR(I)
2570 V=Y+DIR(J)
2580 IF U=X THEN 2590 ELSE 2600
2590 IF V=Y THEN 2790
2600 IF PL=1 THEN 2620
2610 IF A(U,V)=3 THEN 2630 ELSE 2790
2620 IF A(U,V)=4 THEN 2630 ELSE 2790
2630 U=U+DIR(I)
2640 V=V+DIR(J)
2650 IF A(U,V)<=1 THEN 2790
2660 IF A(U,V)=2 THEN 2700
2670 IF PL=1 THEN 2690
2680 IF A(U,V)=4 THEN 2700 ELSE 2630
2690 IF A(U,V)=3 THEN 2700 ELSE 2630
2700 U=X+DIR(I)
2710 V=Y+DIR(J)
2720 IF A(U,V)=PL+2 THEN 2790
2730 EX(EXTRA)=U
2740 EY(EXTRA)=V
2750 EXTRA=EXTRA+1
2760 U=U+DIR(I)
2770 V=V+DIR(J)
2780 GO TO 2720
2790 NEXT J
2800 NEXT I
2810 EXTRA=EXTRA-1
2820 RETURN
2830 REM COLOR ADDITION SQUARES
2840 XX=X
2850 YY=Y
2860 FOR K=1 TO EXTRA
2870 X=EX(K)
2880 Y=EY(K)
2890 GOSUB 2270
2900 A(X,Y)=PL+2
2910 NEXT K
2920 X=XX
2930 Y=YY
2940 RETURN
2950 REM GET COMPUTER MOVE
2960 EXTRAP=-100
2970 FOR X=1 TO 8
2980 FOR Y=1 TO 8
2990 IF A(X,Y)<>0 THEN 4110
3000 GOSUB 2550
3010 IF EXTRA=0 THEN 4110
3020 IF TURN<57 THEN 3040
3030 HARD=1
3040 IF HARD=1 THEN 4070
3050 IF X=1 THEN 3100
3060 IF Y=1 THEN 3080
3070 IF A(X-1,Y-1)=PL+2 THEN 3120
3080 IF Y=8 THEN 3100
3090 IF A(X-1,Y+1)=PL+2 THEN 3120
3100 IF X=8 THEN 3140
3110 IF Y=1 THEN 3130
3120 IF A(X+1,Y-1)=PL+2 THEN 4070
3130 IF Y=8 THEN 3140
3140 IF X=2 THEN 3160
3150 IF Y=2 THEN 4070
3160 IF X=7 THEN 3180
3170 IF Y=7 THEN 4070
3180 IF X*X+Y*Y+56<>9*(X+Y) THEN 3200
3190 EXTRA=EXTRA+1.1
3200 EXTRA=EXTRA+4
3210 GO TO 4070
```

```
3120 IF Y=2 THEN 3210
3190 IF Y=7 THEN 3210
3200 GOTO 4070
3210 U=(7*X-9)/5
3220 V=(7*Y-9)/5
3230 IF A(U,V)>1 THEN 4070
3240 EXTRA=EXTRA-12
3250 GOTO 4070
3260 IF HARD=2 THEN 3360
3270 FOR I=1 TO 8
3280 EDGE(I)=A(I,Y)
3290 NEXT I
3300 FOR I=1 TO EXTRA
3310 IF EY(I)<>Y THEN 3330
3320 EDGE(EX(I))=PL+2
3330 NEXT I
3340 Z=X
3350 GOTO 3570
3360 EXTRA=EXTRA+5
3370 GOTO 4070
3380 IF Y=1 THEN 3520
3390 IF Y=8 THEN 3520
3400 IF HARD=2 THEN 3500
3410 FOR I=1 TO 8
3420 EDGE(I)=A(X,I)
3430 NEXT I
3440 FOR I=1 TO EXTRA
3450 IF EX(I)<>X THEN 3470
3460 EDGE(EY(I))=PL+2
3470 NEXT I
3480 Z=Y
3490 GOTO 3570
3500 EXTRA=EXTRA+5
3510 GOTO 4070
3520 IF HARD=2 THEN 3530
3530 EXTRA=EXTRA+14
3540 GOTO 4070
3550 EXTRA=EXTRA+6
3560 GOTO 4070
3570 CORN=0
3580 FOR I=1 TO EXTRA
3590 IF 2*(EX(I)-2)*(EX(I)-7)+(EY(I)-2)
*(EY(I)-7)=0 THEN 3610
3600 GOTO 3660
3610 U=(7*EX(I)-9)/5
3620 V=(7*EY(I)-9)/5
3630 IF A(U,V)<=1 THEN 3650
3640 GOTO 3660
3650 CORN=1
3660 NEXT I
3670 TYPE=0
3680 IF EDGE(1)=PL+2 THEN 3730
3690 LT=EDGE(1)
3700 IF LT<>0 THEN 3740
3710 LT=1
3720 GOTO 3740
3730 LT=2
3740 FOR I=2 TO Z-1
3750 IF EDGE(I)=PL+2 THEN 3800
3760 IF EDGE(I)<>LT THEN 3790
3770 LT=EDGE(I)
3780 GOTO 3800
3790 LT=1
3800 NEXT I
3810 IF LT=2 THEN 3970
3820 IF EDGE(8)=PL+2 THEN 3870
3830 HT=EDGE(8)
3840 IF HT<>0 THEN 3880
3850 HT=1
3860 GOTO 3880
3870 HT=2
3880 FOR I=7 TO Z+1 STEP -1
3890 IF EDGE(I)=PL+2 THEN 3940
3900 IF EDGE(I)<=1 THEN 3930
3910 HT=EDGE(I)
3920 GOTO 3940
3930 HT=1
3940 NEXT I
3950 IF HT=2 THEN 3970
3960 IF HT<>LT THEN 4030
3970 TYPE=1
3980 IF CORN=0 THEN 4010
3990 EXTRA=EXTRA-8
4000 GOTO 4070
4010 EXTRA=EXTRA+8
4020 GOTO 4070
4030 IF CORN=0 THEN 4060
4040 EXTRA=EXTRA-12
4050 GOTO 4070
4060 EXTRA=EXTRA-4
4070 IF EXTRA>EXTRAP THEN 4080 ELSE 411
0
4080 EXTRAP=EXTRA
4090 ZX=X
4100 ZY=Y
4110 NEXT Y
4120 NEXT X
4130 IF EXTRAP=-100 THEN 4140 ELSE 4210
4140 FOR X=1 TO 8
4150 FOR Y=1 TO 8
4160 IF A(X,Y)>1 THEN 4190
4170 ZX=X
4180 ZY=Y
4190 NEXT Y
4200 NEXT X
4210 X=ZX
4220 Y=ZY
4230 RETURN
4240 END
```

San Francisco Tourist

TI BASIC



"I left my heart in San Francisco . . ." Designed to highlight the sights that abound in and around the City by the Bay, this TI BASIC program is actually two games in one.

First, try your skill at driving down Lombard Street between Hyde and Leavenworth. It's on a steep hill and is known as the "crookedest street in the world." Use the left and right arrow keys (S and D) to steer down the red brick road without bumping into the white concrete sides—or onto someone's green lawn.

Now drive north across the Golden Gate Bridge to Muir Woods, a beautiful, peaceful forest with some of the world's tallest living trees. Start at the upper left corner of the screen and take a quiet walking tour through the woods. Use the arrow keys to change direction, then press ENTER to mark the trees you've seen on your map.

Programming Techniques

This game program implements many of the features discussed in the article *Fun and Games*. The title screen presents the choice of games, and the player need only press the key of his choice (wrong keys are ignored). The program will then branch to the appropriate game, and a screen of instructions is printed. The screen stays on the instructions only as long as the player wishes. The player can just press any key when he is ready to start the game.

Crookedest Street uses scrolling during printing to simulate the road going past. A DEFinition statement near the beginning of the program on line 170 defines a random coordinate R between −3 and +3. A line of road is printed offset R from the previous line. Lines 820 to 870 make sure the road line stays on the screen.

Both games move an object (represented by one graphics character for simplicity and speed) by using the arrow keys. In *Crookedest Street* only the left and right arrow keys are used. The car is always drawn on Row 7, and the arrow keys determine whether the car is drawn in the same column, two columns to the left, or two columns to the right. Lines 930-980 keep the car on the screen. In *Muir Woods* the person may move up, down, left, or right, but will not wrap—staying at the edge, instead. The person will also continue to move in one direction until another arrow key is pressed; the character is moved in each CALL KEY loop.

CALL GCHAR(X,Y,G) is used in both games. In *Crookedest Street* you need to know if the new position of the car is a red square (okay), a white square (crash), or a green square (fatal crash). After the new car position is drawn, the old position must be replaced by the appropriately colored square.

Muir Woods uses GCHAR to determine positions of trees for marking. Also, the person leaves a trail. So if the square was a blank, the trail is printed; but if it was a tree or a marked tree, that character stays there.

Muir Woods also demonstrates the use of a timer or counter in the CALL KEY loop. You may change the value 100 for SH in line 1910 for more or less time.

I wanted to use [ENTER] as the key to press for "firing," so the split keyboard method of detecting the "fire" key was not possible. If you use the split keyboard you can alternate calling the halves of the keyboard and detect the "fire" key sooner, but since the codes are different for the 99/4 and the 99/4A consoles, the game instructions would have to be different. [ENTER] is not detected on the 99/4A; you must press the period to return key code 13. In these games the quickest way to detect [ENTER] is to let go of the arrow keys before pressing [ENTER].

EXPLANATION OF THE PROGRAM

San Francisco Tourist

Line Nos.	
150-170	Defines functions to be used as random coordinates.
180-240	Clears screen; defines graphics characters for bridge.
250-380	Prints bridge and title; if the program is just starting, plays "I Left My Heart in San Francisco."
390	Prints choices of games.
400-420	Defines graphics characters for games.

430-460	Receives player's input and branches appropriately.
470-500	Subroutine to press any key to start.
510-530	Subroutine to delay.
540-610	Prints instructions for *Crookedest Street*; defines graphics characters; waits for player to press a key.
620-750	Clears screen; defines graphics colors; prints game screen. DATA contains coordinates for printing road.
760-780	Initializes coordinates of road and car.
790-860	Prints 75 lines of crooked street randomly; last 15 lines are straight.
870-980	Makes sound, draws new position of car depending on key pressed; replaces old position with proper graphics character.
990-1070	Tests for crash; makes a sound and increments number of crashes.
1080-1170	Ending remarks; plays victory melody for zero crashes.
1180-1210	Procedure if car goes into green.
1220-1250	Delays, then waits for player to press a key.
1260-1290	Clears screen; returns colors to black; branches to menu screen.
1300-1400	Prints instructions for *Muir Woods* and defines graphics characters and colors.
1410-1470	Clears screen, randomly draws 70 trees on screen.
1480-1520	Initializes time, marked trees, coordinates, graphics.
1530-1540	Places person at entrance and sounds initial beep.
1550-1860	Moves person depending on key pressed. Person will not "wrap" but stays at edge.
1870-1920	Increments time and prints time; if time = 100, ends game.
1930-2010	Procedure for marking tree.
2020-2100	Ending statements; returns colors to black; branches to menu screen.
2110-2120	Ends program.

```
100 REM  *********************
110 REM  *  SF TOURIST  *
120 REM  *********************
130 REM
140 REM
150 DEF FNR(X)=INT(10*RND(X))+1
160 DEF FNC(X)=INT(26*RND(X))+2
170 DEF FNA(X)=(-1)^(INT(4*RND(X)))*(INT(4*RND(X))+1)
180 CALL CLEAR
190 CALL CHAR(64,"18183C3C5A5A9999")
200 CALL CHAR(35,"0101020408307FFF")
210 CALL CHAR(47,"1818181818FFFF")
220 CALL CHAR(36,"8060180607FFFF")
230 CALL CHAR(37,"0106186060FFFF")
240 CALL CHAR(38,"8080402018060FFFF")
250 CALL SCREEN(4)
260 PRINT TAB(12);"@  @":TAB(11);"#/$%
/&"
270 PRINT :::::"** SAN FRANCISCO TOURIST
 *":::::
280 IF I<>0 THEN 390
290 P=300
300 CALL SOUND(P,330,0)
310 CALL SOUND(P,349,0)
320 CALL SOUND(P,440,0)
330 CALL SOUND(4*P,392,0,131,10,165,8)
340 CALL SOUND(P,440,0)
350 CALL SOUND(P,494,0,165,10,196,8)
360 CALL SOUND(P,523,0)
370 CALL SOUND(P,440,0,147,10)
380 CALL SOUND(4*P,294,0,220,8,175,10)
390 PRINT "WHICH DO YOU WISH TO VISIT?
":::"  1  CROOKEDEST STREET":::"  2
  MUIR WOODS":::"  3  END PROGRAM":
::
400 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
410 CALL CHAR(41,"FFFFFFFFFFFFFFFF")
420 CALL CHAR(48,"0")
430 CALL KEY(0,K,S)
440 IF (K<49)+(K>51)=-1 THEN 430
450 CALL CLEAR
460 ON K-48 GOTO 540,1300,2110
470 PRINT "PRESS ANY KEY TO START.";
480 CALL KEY(0,K,S)
490 IF S<1 THEN 480
500 RETURN
510 FOR I=1 TO 500
520 NEXT I
530 RETURN
540 PRINT "  **  CROOKEDEST STREET  **"
550 CALL CHAR(96,"307C7C5858787C7C58")
560 CALL CHAR(97,"C636361818363663C3")
570 PRINT :::"LOMBARD STREET IS THE":::"
CROOKEDEST STREET IN THE":::"WORLD.
   IT IS ONE BLOCK"
580 PRINT :"LONG ON A STEEP HILL.":::"Y
OUR CHALLENGE IS TO DRIVE":::"DOWN
THE RED BRICK ROAD"
590 PRINT :"WITHOUT BUMPING THE CONCRE
TE":"TO SIDES.  USE THE ARROW KEYS":
:"TO STEER.":::
600 C=0
610 GOSUB 470
620 DATA 6,6,7,8,9,11,13,15,18,20,20,1
7,15,13,11,9,8,6,6,8,11,13,16,19,20
630 CALL CLEAR
640 CALL SCREEN(5)
650 CALL COLOR(2,7,16)
660 CALL COLOR(9,2,11)
670 CALL COLOR(1,12,3)
680 RESTORE 620
690 FOR I=1 TO 24
700 READ J
710 CALL HCHAR(1,J,40,8)
720 CALL HCHAR(1,J+1,41,6)
730 NEXT I
740 CALL HCHAR(7,17,96)
750 RANDOMIZE
760 I=18
770 X=17
780 G=41
790 FOR I=1 TO 75
800 IF J>59 THEN 860
810 J=J+R
820 IF J<21 THEN 840
830 J=21
840 IF J>1 THEN 860
850 J=1
860 PRINT TAB(J);"((((((((("
870 CALL SOUND(-100,110,1,-1,1)
880 CALL HCHAR(6,X,G)
890 CALL KEY(0,K,S)
900 IF (K<>83)+(K<>68)=-2 THEN 990
910 IF K=83 THEN 960
920 X=X+2
930 IF X<31 THEN 990
940 X=31
950 GOTO 990
960 X=X-2
970 IF X>2 THEN 990
980 X=2
990 CALL GCHAR(7,X,G)
1000 IF G=41 THEN 1060
1010 IF G=96 THEN 1060
1020 IF G=32 THEN 1120
1030 CALL SOUND(-50,-5,0)
1040 CALL HCHAR(7,X,97)
1050 C=C+1
1060 CALL HCHAR(7,X,96)
1070 NEXT I
1080 CALL HCHAR(22,1,32,64)
```

```
1090 PRINT "YOU MADE IT";"!!!"NUMBER OF C
     RASHES:";C
1100 IF C>0 THEN 1220
1110 DATA 330,392,523,659,523,659,659
1120 RESTORE 1110
1130 FOR I=1 TO 7
1140 READ S
1150 CALL SOUND(150,S,0)
1160 NEXT I
1170 GOTO 1220
1180 CALL SOUND(200,-5,0,400,0)
1190 CALL HCHAR(7,X,97,2)
1200 CALL HCHAR(22,1,32,64)
1210 PRINT "SORRY; THE CAR IS DAMAGED";
     :"BEYOND REPAIR"
1220 GOSUB 510
1230 PRINT :::"PRESS ANY KEY"
1240 CALL KEY(0,K,S)
1250 IF S<1 THEN 1240
1260 CALL CLEAR
1270 CALL COLOR(2,2,1)
1280 CALL COLOR(1,2,1)
1290 GOTO 250
1300 PRINT TAB(6);"** MUIR WOODS **"
1310 CALL CHAR(96,"1C1C887F081422")
1320 CALL CHAR(97,"10107C101")
1330 CALL COLOR(9,7,1)
1340 CALL CHAR(104,"1031107C10FE101")
1350 CALL CHAR(105,"2FF81A5999A581FF")
1360 CALL COLOR(10,13,1)
1370 PRINT :::"MUIR WOODS IS A BEAUTIFUL
     ":::"FOREST OF GIANT TREES"
1380 PRINT :"NORTH OF SAN FRANCISCO.":::
     "TAKE A TOUR AND MARK AS":::"MANY T
     REES ON YOUR MAP AS"
1390 PRINT :"YOU CAN. MOVE BY PRESSING
     ":::"THE ARROW KEYS; MARK TREES":::"
     BY PRESSING <ENTER>.":::::
1400 GOSUB 470
1410 CALL CLEAR
1420 CALL SCREEN(12)
1430 PRINT "TIME":::"TREES"
1440 RANDOMIZE
1450 FOR I=1 TO 70
1460 CALL HCHAR(R19,R28,104)
1470 NEXT I
1480 SH=0
1490 P=0
1500 X=2
1510 Y=2
1520 G=32
1530 CALL HCHAR(2,2,96)
1540 CALL SOUND(150,1397,4)
1550 CALL KEY(0,K,S)
1560 IF K=13 THEN 1930
1570 IF K<>69 THEN 1610
1580 DX=-1
1590 DY=0
1600 GOTO 1720
1610 IF K<>68 THEN 1650
1620 DX=0
1630 DY=1
1640 GOTO 1720
1650 IF K<>88 THEN 1690
1660 DX=1
1670 DY=0
1680 GOTO 1720
1690 IF K<>83 THEN 1720
1700 DX=0
1710 DY=-1
1720 IF G>-104 THEN 1740
1730 G=97
1740 CALL HCHAR(X,Y,G)
1750 X=X+DX
1760 IF X>1 THEN 1780
1770 X=1
1780 IF X<20 THEN 1800
1790 X=20
1800 Y=Y+DY
1810 IF Y>2 THEN 1830
1820 Y=2
1830 IF Y<31 THEN 1850
1840 Y=31
1850 CALL GCHAR(X,Y,G)
1860 CALL HCHAR(X,Y,96)
1870 SH=SH+1
1880 FOR I=1 TO LEN(STR$(SH))
1890 CALL HCHAR(21,I+9,ASC(SEG$(STR$(S
     H),I,1)))
1900 NEXT I
1910 IF SH=100 THEN 2020
1920 GOTO 1550
1930 CALL SOUND(100,-2,0)
1940 IF G<>104 THEN 1870
1950 CALL HCHAR(X,Y,105)
1960 G=105
1970 P=P+1
1980 FOR I=1 TO LEN(STR$(P))
1990 CALL HCHAR(23,I+9,ASC(SEG$(STR$(P
     ),I,1)))
2000 NEXT I
2010 GOTO 1870
2020 PRINT :"TIME IS UP.":::"YOU LOCKED
     AT";P;"TREES."
2030 GOSUB 510
2040 PRINT :::"PRESS ANY KEY"
2050 CALL KEY(0,K,S)
2060 IF S<1 THEN 2050
2070 CALL CLEAR
2080 CALL COLOR(9,2,1)
2090 CALL COLOR(10,2,1)
2100 GOTO 250
2110 CALL CHAR(33,"0101010101010001")
2120 END
```

TI BASIC

COUNTY FAIR DERBY

County Fair Derby is a party game in which up to eight players bet on horses in a color-animated race. Our family finds it quite exciting—especially with three or more players. There is, however, only one keyboard operator; the rest is up to the computer. In addition to running the five horses, the computer keeps tabs on each horse's track record, plus the bankroll of each player. The program operation is simple and self-prompting. To break the input loop, the word LAST must be entered. If this word is misspelled, it then becomes just another player's name. When this TI BASIC program was loaded into Extended BASIC for the purpose of checking available memory left, the SIZE command revealed that there were 4873 bytes left. This leaves enough memory for you to add to, or use to modify the program. You might try giving the computer a fixed amount of money before the races start and having the players try to "break the track." Other bells and whistles I leave to your imagination. Here's hoping you enjoy the program as much as I enjoyed writing it. But don't waste another minute. It's already post time—the horses will soon be off and running . . . .

EXPLANATION OF THE PROGRAM
County Fair Derby

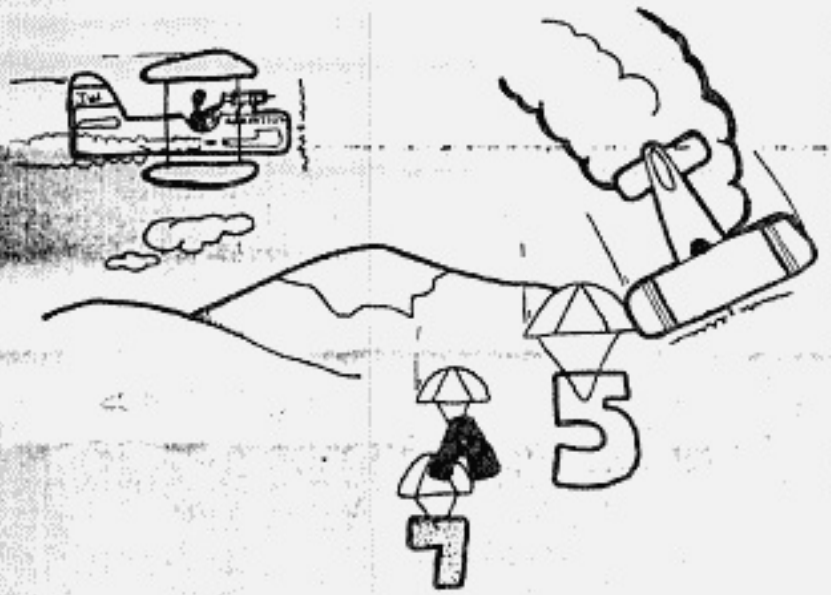
Line Nos.	
140-340	Introduction display and odds table.
350-420	Introductory music and wait for key.
430-1060	Initialization and define characters to be used for display.
1070-1560	Input routines: players' names, choices for horse selection, kind of bet and amount. Typing LAST for player's name breaks the INPUT loop.
1570-1810	Draws track with lane numbers and plays post-time tune.
1820	Z is a switch to control RETURN from subroutine at 2490.
1830-2020	Positions horses on the track in the proper place and color (subroutine at 2490 draws horse and RETURNs if Z equals 1).
2030-2190	Rests Z; sets starting coordinates for horses (K and S are variables used later in determing win, place and show.) Waits for "S" key to start.
2200-2460	Generates random number from one to five to determine which horse to move. Line number 2220 (ON N GOTO) finds position of horse, sets coordinates for move routine and jumps to move routine. (If the vertical coordinate has been set to zero, the horse has finished and the program jumps back for a new random number.)
2470-2600	Moves horse through an animation loop and redraws it two positions forward from where it started. ("Q" is used as a control switch to pass through the loop twice.)
2610-2830	Checks if V>28 (end of race for that horse); if not, sets new coordinate values and jumps back for new random number.
2840-3120	Calculates the finishing horse (D). If S equals 0, the horse wins. Set S equals winning number. Line 2870 (ON S GOSUB) sets color for winning announcement. Line 2990 (ON S GOTO) sets column to zero to remove horse from race; jumps back for a new random number and continues. If S<>0, then the finishing horse becomes K for second place and, (except for setting color) a similar routine is followed. If K<>0, then D becomes the third place horse and the race stops.
3130-3510	Displays win, place, show announcement and waits for key.
3520-3870	On K1(X) goes to the kind of bet player (X) made. Checks to see if player (X) has won and calculates the amount. If there are winnings, goes to subroutine 4090. For no winnings, GOSUB 3970. On return goes to 3880.
3880-3960	Increments (X). Checks to see if four results have been displayed; if so, goes to 4130 and waits for key before returning for next results (3550); if not over four, goes directly to 3550.
3970-4010	Subroutine to update and display losers.
4020-4080	Subroutine to update and display losers in debt.
4090-4120	Subroutine to update and display winners.
4130-4160	Wait for key and check for LAST before continuing.
4170-4290	Update past records and display for players betting on trends. Wait for key.
4300-4340	Loop back for INPUTS of next race.
4350-4380	Data for music. Use "break" key to end program.

```
100 REM  *********************
110 REM  *  COUNTY FAIR DERBY  *
120 REM  *********************
130 REM
135 REM
140 CALL CLEAR
150 CALL COLOR(2,2,14)
160 FOR I=3 TO 8
170 CALL COLOR(I,2,12)
180 NEXT I
190 CALL HCHAR(24,2,42,28)
200 PRINT
210 PRINT TAB(8);"COUNTY FAIR DERBY":::
220 PRINT TAB(8);"A FIVE HORSE RACE":::
230 PRINT TAB(4);"YOU CAN BET FOUR WAY
    S":::
240 PRINT "<1> WIN PAYS 4 TO 1":::
250 PRINT "<2> PLACE PAYS 2 TO 1":::
260 PRINT "<3> SHOW PAYS 2 TO 1":::
270 PRINT "<4> PARLAY PAYS 15 TO 1"
    :::
280 CALL HCHAR(24,9,42,14)
290 PRINT :::
300 PRINT "PARLAY<PICK 1ST; AND 2ND;>
    ":::
310 PRINT "EACH PLAYER IS GIVEN $200"::
    :
320 CALL HCHAR(24,2,42,28)
330 CALL VCHAR(1,2,42,24)
340 CALL VCHAR(1,30,42,24)
350 RESTORE 4370
360 READ DU,NO
370 IF DU=0 THEN 400
380 CALL SOUND(500*DU,NO,5)
390 GOTO 360
400 PRINT "PRESS ANY KEY"
410 CALL KEY(0,KEY,STAT)
420 IF STAT=0 THEN 410
430 CALL CLEAR
440 PRINT TAB(7);"***BANG ON***":::::
450 PRINT TAB(7);"GOTTA GET THE":::::
460 PRINT TAB(11);"HORSES":::::::
470 DIM HS(50)
480 HS(1)="0000000004020106F"
490 HS(2)="00000E080F1F3378"
500 HS(3)="0F0F1020400000000"
510 HS(4)="F0F0080402080000"
520 HS(5)="0000000000000007F"
530 HS(6)="0000000006001E5EF0"
540 HS(7)="0F0F0808040200000"
550 HS(8)="F0F0102010400000000"
560 HS(9)="0000000103870101"
570 HS(10)="00000F1F3000000007"
580 HS(11)="0000F1F5F3100000003"
590 HS(12)="0000000000010306C"
600 HS(13)="00007070606070707"
610 HS(14)="0000500808080808080"
620 HS(15)="0000C0C0606060606"
630 HS(16)="0000C0E0F0707070E0"
640 HS(17)="0000E0E0E0606060"
650 HS(18)="0000F0F0000C0F0"
660 HS(19)="0101010707000000"
670 HS(20)="1F1F3030303F3F0000"
680 HS(21)="0300003131F1F0000"
690 HS(22)="0F0F0000000000000"
700 HS(23)="0000000607301000000"
710 HS(24)="0000808080E0E0000000"
720 HS(25)="0000000E0E0000000"
730 HS(26)="E0707070E0C0000000"
740 HS(27)="F1F1606060000000"
750 HS(28)="7111111F0E000000"
760 HS(29)="0000503830303131"
770 HS(30)="0000C0C0C0C0C0C0"
780 HS(31)="C3C181818181818"
790 HS(32)="5030305C3E3735"
800 HS(33)="C0C0C0C0C0C0"
810 HS(34)="0000F3F3030303F"
820 HS(35)="0000FCFC0C0C0C0C"
830 HS(36)="1F1F1F1E1C1C0000"
840 HS(37)="FBFBFBFB38380000"
850 HS(38)="1818181C3C3C0000"
860 HS(39)="5151515030303000"
870 HS(40)="CCEC7C3C1C0C0000"
880 HS(41)="5F8080303737F000"
890 HS(42)="FC0C0C0C7CFC0000"
900 D=129
910 K=1
920 FOR D=D TO D+7
930 CALL CHAR(D,HS(K))
940 K=K+1
950 NEXT D
960 IF D>152 THEN 980
970 GOTO 910
980 CALL CLEAR
990 CALL COLOR(11,15,8)
1000 CALL COLOR(12,14,11)
1010 CALL COLOR(13,13,11)
1020 CALL COLOR(14,2,11)
1030 CALL COLOR(15,7,11)
1040 CALL COLOR(16,5,11)
1050 CALL COLOR(2,2,12)
1060 CALL CLEAR
1070 X=1
1080 CALL CLEAR
1090 PRINT "TYPE PLAYER'S NAME ?":::
1100 PRINT "AFTER THE LAST PLAYER'S NAM
    E":::
1110 PRINT "HAS BEEN ENTERED TYPE LAST"
    :::
1120 INPUT "NAME ?":NAME$(X)
1130 IF NAME$(X)="LAST" THEN 1570
1140 IF X>=8 THEN 1160
1150 GOTO 1190
1160 PRINT "EIGHT IS THE MAX.NUM.OF PLA
    YERS"
1170 PRINT "TYPE LAST TO CONTINUE"
1180 GOTO 1120
1190 TOT(X)=200
1200 CALL CLEAR
1210 GOSUB 1230
1220 GOTO 1080
1230 PRINT :"O.K. ";NAME$(X);" PICK A H
    ORSE ?":::
1240 INPUT "HORSE ?":HO(X)
1250 IF HO(X)>5 THEN 1270
1260 GOTO 1310
1270 GOSUB 1290
1280 GOTO 1240
1290 PRINT :"NUM. TOO BIG TRY AGAIN":::
1300 RETURN
1310 PRINT :"WHAT KIND OF BET ?<1 TO 4>
    ":::
1320 PRINT "<1>= WIN":::
1330 PRINT "<2>= PLACE":::
1340 PRINT "<3>= SHOW":::
1350 PRINT "<4>= PARLAY":::
1360 INPUT "KIND ?":KI(X)
1370 IF KI(X)>4 THEN 1400
1380 IF KI(X)=4 THEN 1500
1390 GOTO 1420
1400 GOSUB 1290
1410 GO TO 1560
1420 PRINT :"HOW MUCH DO YOU BET ?<$1 T
    O $200>":::
1430 INPUT "BET ?":BET(X)
1440 IF BET(X)>200 THEN 1400
1450 GOTO 1490
1460 GOSUB 1290
1470 GOTO 1430
1480 X=X+1
1490 RETURN
1500 PRINT :"YOU PICKED NO.";HO(X);"  T
    O WIN":::
1510 PRINT "WHICH HORSE TO PLACE ?":::
1520 INPUT "PLACE ?":PA2(X)
1530 IF PA2(X)>5 THEN 1550
1540 GOTO 1420
1550 GOSUB 1290
```

```
1560 GOTO 1520
1570 CALL CLEAR
1580 PRINT "PRESS S TO START"
1590 CALL COLOR(2,11,11)
1600 FOR X=1 TO 22
1610 PRINT
1620 NEXT X
1630 CALL CHAR(119,"81C3663C183C6681")
1640 CALL HCHAR(9,1,119,30)
1650 CALL HCHAR(20,1,119,30)
1660 X=10
1670 T=2
1680 FOR A=1 TO 10
1690 CALL HCHAR(X,T,42,29)
1700 X=X+1
1710 NEXT A
1720 RESTORE 4350
1730 READ DU,NO                    IF DU=0 THEN 1720
1750 CALL SOUND(200+DU,NO,5)
1760 GOTO 1730
1770 CALL HCHAR(10,2,49)
1780 CALL HCHAR(12,2,50)
1790 CALL HCHAR(14,2,51)
1800 CALL HCHAR(16,2,52)
1810 CALL HCHAR(18,2,53)
1820 Z=1
1830 D=120
1840 R=10
1850 V=3
1860 GOSUB 2490
1870 D=126
1880 R=12
1890 V=3
1900 GOSUB 2490
1910 D=136
1920 R=14
1930 V=3
1940 GOSUB 2490
1950 D=144
1960 R=16
1970 V=3
1980 GOSUB 2490
1990 D=152
2000 R=18
2010 V=3
2020 GOSUB 2490
2030 Z=0
2040 A=10
2050 B=14
2060 I=16
2070 J=18
2080 L=12
2090 F=14
2100 O=10
2110 P=14
2120 G=14
2130 H=0
2140 E=0
2150 S=0
2160 CALL KEY(0,KEY,STATUS)
2170 IF STATUS=0 THEN 2160
2180 IF KEY=83 THEN 2200
2190 GOTO 2150
2200 RANDOMIZE
2210 N=INT(5*RND)+1
2220 ON N GOTO 2250,2280,2330,2380,2430
2230 E=A
2240 V=B
2250 IF E=0 THEN 2200
2260 D=120
2270 GOTO 2470
2280 R=E
2290 V=F
2300 IF F=0 THEN 2200
2310 D=128
2320 GOTO 2470
2330 E=G
2340 V=H
2350 IF H=0 THEN 2200
2360 D=130
2370 GOTO 2470
2380 R=I
2390 V=J
2400 IF J=0 THEN 2200
2410 D=144
2420 GOTO 2470
2430 R=O
2440 V=P
2450 D=152
2460 IF P=0 THEN 2200
2470 CALL HCHAR(R,V-1,42)
2480 CALL HCHAR(R+1,V-1,42)
2490 CALL HCHAR(R,V,D)
2500 CALL HCHAR(R,V+1,D+1)
2510 CALL HCHAR(R+1,V,D+2)
2520 CALL HCHAR(R+1,V+1,D+3)
2530 CALL SOUND(200,220,2)
2540 IF E=0 THEN 2550
2550 RETURN
2560 IF Q=1 THEN 2610
2570 Q=1
2580 V=V+1
2590 D=D+4
2600 GOTO 2470
2610 D=D-4
2620 Q=0
2630 IF V>28 THEN 2840
2640 V=V+1
2650 IF D=120 THEN 2720
2660 IF D=128 THEN 2750
2670 IF D=144 THEN 2780
2680 IF D=152 THEN 2810
2690 G=R
2700 H=V
2710 GOTO 2200
2720 A=R
2730 B=V
2740 GOTO 2200
2750 E=R
2760 F=V
2770 GOTO 2200
2780 I=R
2790 J=V
2800 GOTO 2200
2810 O=R
2820 P=V
2830 GOTO 2200
2840 D=(D-112)/8
2850 IF S<>0 THEN 3000
2860 S=D
2870 ON S GOSUB 2890,2910,2930,2950,297
     0
2880 GOTO 2990
2890 CALL COLOR(9,2,14)
2900 RETURN
2910 CALL COLOR(9,15,15)
2920 RETURN
2930 CALL COLOR(9,15,2)
2940 RETURN
2950 CALL COLOR(9,2,7)
2960 RETURN
2970 CALL COLOR(9,2,5)
2980 RETURN
2990 ON S GOTO 3030,3050,3070,3090,3110
3000 IF K<>0 THEN 3150
3010 K=D
3020 ON K GOTO 3030,3050,3070,3090,3110
3030 B=0
3040 GOTO 2200
3050 F=0
3060 GOTO 2200
3070 H=0
3080 GOTO 2200
3090 J=0
3100 GOTO 2200
3110 P=0
3120 GOTO 2200
```

```
3150 R=22
3160 V=15+8*T
3170 FOR T=1 TO 4
3180 CALL CHAR(95+T,H$(X))
3190 X=X+5
3200 NEXT T
3210 FOR T=1 TO 2
3220 CALL HCHAR(R,V,95+T)
3230 V=V+1
3240 NEXT T
3250 V=V-2
3260 R=R+1
3270 FOR T=3 TO 4
3280 CALL HCHAR(R,V,95+T)
3290 V=V+1
3300 NEXT T
3310 CALL COLOR(10,15,6)
3320 CALL COLOR(11,15,6)
3330 R=22
3340 V=15
3350 Q=11
3360 T=11
3370 FOR T=T TO T+6
3380 CALL CHAR(103+Y,H$(20+T))
3390 CALL HCHAR(R,V,103+Y)
3400 V=V+1
3410 NEXT T
3420 IF Q=0 THEN 3470
3430 R=23
3440 V=11
3450 Q=0
3460 GOTO 3360
3470 PRINT :TAB(7);X;" PLACES"
3480 PRINT :TAB(7);D;" SHOWS"::
3490 PRINT "PRESS ANY KEY"
3500 CALL KEY(0,K,ST)
3510 IF ST=0 THEN 3500
3520 CALL COLOR(2,2,12)
3530 CALL CLEAR
3540 X=1
3550 IF NAME$(X)=" LAST" THEN 4130
3560 ON X GOTO 3570,3640,3720,3810
3570 IF HO(X)=5 THEN 3600
3580 GOSUB 5970
3590 GOTO 3880
3600 BET(X)=BET(X)+4
3610 BET(X)=INT(BET(X)*100+.5)/100
3620 GOSUB 4090
3630 GOTO 3880
3640 IF HO(X)=5 THEN 3680
3650 IF HO(X)=K THEN 3680
3660 GOSUB 5970
3670 GOTO 3880
3680 BET(X)=BET(X)*3/2
3690 BET(X)=INT(BET(X)*100+.5)/100
3700 GOSUB 4090
3710 GOTO 3880
3720 IF HO(X)=5 THEN 3770
3730 IF HO(X)=K THEN 3770
3740 IF HO(X)=D THEN 3770
3750 GOSUB 5970
3760 GOTO 3850
3770 BET(X)=BET(X)*2/3
3780 BET(X)=INT(BET(X)*100+.5)/100
3790 GOSUB 4090
3800 GOTO 3880
3810 IF HO(X)<>5 THEN 3830
3820 IF PA2(X)=K THEN 3850
3830 GOSUB 5970
3840 GOTO 3880
3850 BET(X)=BET(X)*1.5
3860 BET(X)=INT(BET(X)*100+.5)/100
3870 GOSUB 4090
3880 X=X+1
3890 IF X>5 THEN 3550
3900 IF X>4 THEN 3920
3910 GOTO 3530
3920 GOTO 4130
3930 CALL CLEAR
3940 GOTO 3550
3950 IF X<=8 THEN 3550
3960 GOTO 3930
3970 IF TOT(X)<BET(X)THEN 4020
3980 PRINT "SO SORRY ";NAME$(X);" YOU L
OST";BET(X)::
3990 TOT(X)=TOT(X)-BET(X)
4000 PRINT "YOU NOW HAVE $";TOT(X)::
4010 RETURN
4020 TOT(X)=TOT(X)*.1
4030 PRINT "HEY ";NAME$(X);" YOU LOSE
AGAIN.":::
4040 TOT(X)=TOT(X)-BET(X)
4050 TOT(X)=TOT(X)*.1
4060 PRINT "YOU OWE THE TRACK $";TOT(X
)::::
4070 PRINT "WE HOPE YOUR CREDIT IS GOOD
.":::
4080 RETURN
4090 TOT(X)=TOT(X)+BET(X)
4100 PRINT "GREAT ";NAME$(X);" YOU WIN
$";BET(X)::
4110 PRINT "YOU NOW HAVE $";TOT(X)::
4120 RETURN
4130 PRINT "PRESS ANY KEY"
4140 CALL KEY(0,K,ST)
4150 IF ST=0 THEN 4140
4160 IF NAME$(X)<>" LAST" THEN 3950
4170 CALL CLEAR
4180 L(X)=L(X)+1
4190 D(X)=D(X)+1
4200 W(S)=W(S)+1
4210 PRINT TAB(8);"PAST RECORDS":::
4220 PRINT TAB(1);"NO";"WIN";L(1);"PLA
CE";D(1);"SHOW"
4230 PRINT TAB(2);"NO";W(2);"WIN";L(2);"PL
ACE";D(2);"SHOW"
4240 PRINT TAB(3);"NO";W(3);"WIN";L(3);"PL
ACE";D(3);"SHOW"
4250 PRINT TAB(4);"NO";W(4);"WIN";L(4);"PL
ACE";D(4);"SHOW"
4260 PRINT TAB(5);"NO";W(5);"WIN";L(5);"PL
ACE";D(5);"SHOW"
4270 PRINT :::"PRESS ENTER"
4280 CALL KEY(0,K,ST)
4290 IF ST=0 THEN 4280
4300 CALL CLEAR
4310 X=1
4320 IF NAME$(X)=" LAST" THEN 1570
4330 GOSUB 1250
4340 GOTO 4320
4350 DATA 1,523,1,523,1,523,1,449,1,449
,1,449,1,349,1,449,1,349,2,262
4360 DATA 1,349,1,449,1,523,1,523,1,523
,1,449,1,449,1,262,1,262,1,523
4370 DATA 1,392,1,392,1,330,1,392,1,330
,1,294,1,330,2,262,2,294
4380 DATA 1,392,1,392,1,392,1,330,1,392
,1,440,1,392,2,330,2,294,1,330,1,262
```

EXTENDED BASIC



SPRITE CHASE

Wait . . . Wait . . . Wait . . . When will they get here? Wait . . . Wait . . . Wait . . . "Hi dear, anything in the mail today? Did you look between the doors? Oh. Shucks."

"Hello Ginny. What? You accepted a package from UPS for me? Great! Could you get it for me? Thanks."

"See you later dear, I'll be downstairs."

They're here . . .
The SPRITES are here . . .
NOW, WHAT CAN I DO WITH THEM?

Skim through the manual, page 25. Uh huh. OK. Yeah. This looks great! Let's get a little deeper. Page 64. Oh, oh. Looks like the ALL option of COINC doesn't tell you which SPRITES "coincidenced." I hope someone can find out where to PEEK for this.

Now, what shall I do with them? Something simple. Design some cute characters? No, let's just get those SPRITES moving. Since it might take some time for COINC (ALL,. . .) to figure out which SPRITES are coincidental, I'll stick to one SPRITE versus another. How about a series to chase? Numbers . . . Letters . . . ROTATION . . . That's it . . .

A short game chasing the 10 numbers or a longer game chasing the 26 letters. I'll try the MAGNIFY too. I'll need a numeric variable for the COINC tolerance for that. I guess 8 for normal size and 16 for double size. I'll generate the number or letter SPRITES to go any which way at some speed between −25 and 25. I'll stick to the 8 directions around the arrows for the chaser or else I'll get so tangled up in the math that I'll never move a SPRITE. Wish I had joysticks. I guess some kind of clock would be good for scoring.

Well, here we go!

EXPLANATION OF THE PROGRAM

Sprite Chase

Line Nos.	
170-200	Instructions.
210-280	Set up variations for play.
290-300	Reset for start of game.
310	Make clock numbers reverse image.
320-330	Put the Chaser somewhere in middle of the screen.
340-360	Create the Chasees.
370-390	The chase has begun.
400-450	While waiting for a direction key to be pressed, keep the clock going and check for a coincidence when the Chaser is stationary.
460-530	Start the Chaser in the direction of key pressed.
540-590	While awaiting release of direction key, check for a coincidence when the Chaser is moving; keep clock going.
600-610	Stop the Chaser; wait for another key to be pressed.
620-650	Caught one; go for the next one.
660-710	End of game.
720	That's it.

A FEW POSTSCRIPT NOTES:

If a SPRITE is moving slowly in a vertical direction, it might go off the top or bottom of the screen for a while, but it can be caught there.

If you insert COINC statements between a lot of the instructions and check the HIT field, you probably would reduce the number of times a coincidence is missed.

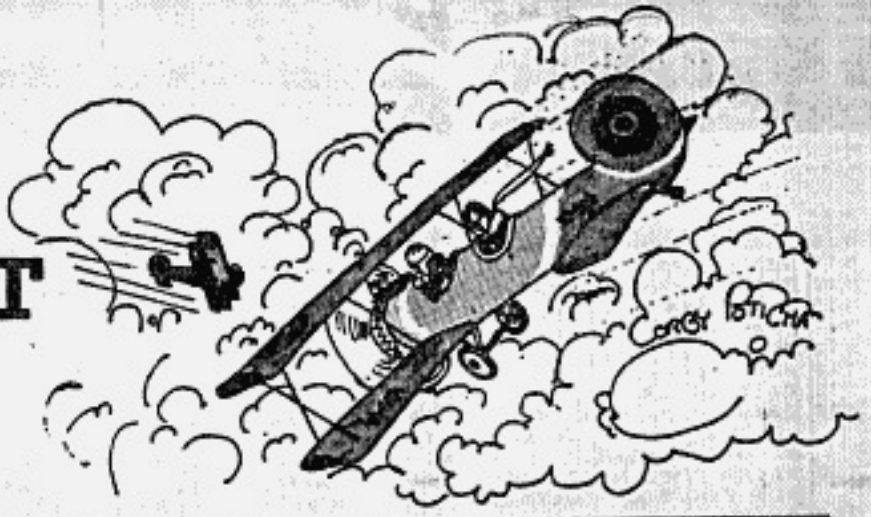
If you leave the Chaser in its original position, all targets will eventually pass through it. I wonder how long this would take?

(If it sounded like I was talking to myself, I was! Doesn't everyone???)

```
100 REM ********************
110 REM *  SPRITE CHASE  *
120 REM ********************
130 REM
140 REM
150 REM
160 REM
170 CALL CLEAR
180 PRINT "USE THE FOUR ARROW KEYS AND
    W,R,Z,C KEYS TO CHASE THE LETTE
    RS OR NUMBERS.":::PRINT
190 PRINT "YOU MUST CATCH THEM IN ALPH
    ABETIC OR NUMERIC SEQUENCE.":::PRINT
200 PRINT "PRESS 'L' FOR LARGE TARGETS
    ,'S' FOR SMALL TARGETS.":::
210 CALL KEY(3,GOT.STATUS)
220 IF STATUS=0 THEN 210
230 IF GOT=76 THEN T=16 ::: CALL MAGNIF
    Y(2) ELSE IF GOT=83 THEN T=8 ELSE 2
    10
240 PRINT "FOR NUMBERS PRESS 'N'.":::"FO
    R LETTERS PRESS 'L'.":::PRINT
250 CALL KEY(0,GOT.STATUS)
260 IF STATUS=0 THEN 250
270 IF GOT=78 THEN TARGS=10 ::: CH=47 E
    LSE IF GOT=76 THEN TARGS=26 ::: CH=
    64 ELSE 250
280 CALL CLEAR
290 RANDOMIZE
300 COUNT=0
310 CALL COLOR(9,2,9)::: CALL COLOR(4,2
    ,9)
320 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
330 CALL SPRITE(#28,96,2,90,120,0,0)
340 FOR A=1 TO TARGS
350 CALL SPRITE(#A,A+CH,2,90,120,INT(R
    ND*50-25),INT(RND*30-25))
360 NEXT A
370 CALL SOUND(100,555,0)
380 FOR A=1 TO TARGS
390 CALL COLOR(#A,16)
400 CALL KEY(0,GOT.STATUS)
410 COUNT=COUNT+1
420 DISPLAY AT(24,1):SIZE(6):COUNT
430 CALL COINC(#28,#A,T,HIT)
440 IF HIT=-1 THEN 620
450 IF STATUS=0 THEN 400
460 IF GOT=69 THEN CALL MOTION(#28,-30
    ,0)::: GOTO 540
470 IF GOT=88 THEN CALL MOTION(#28,30,
    0)::: GOTO 540
480 IF GOT=68 THEN CALL MOTION(#28,0,3
    0)::: GOTO 540
490 IF GOT=83 THEN CALL MOTION(#28,0,-
    30)::: GOTO 540
500 IF GOT=87 THEN CALL MOTION(#28,-30
    ,0)::: GOTO 540
510 IF GOT=82 THEN CALL MOTION(#28,-30
    ,30)::: GOTO 540
520 IF GOT=90 THEN CALL MOTION(#28,30,
    -30)::: GOTO 540
530 IF GOT=67 THEN CALL MOTION(#28,30,
    30)::: GOTO 540
540 CALL KEY(0,GOT.STATUS)
550 CALL COINC(#28,#A,T,HIT)
560 IF HIT=-1 THEN 620
570 COUNT=COUNT+1
580 DISPLAY AT(24,1):SIZE(6):COUNT
590 IF STATUS=-1 THEN 540
600 CALL MOTION(#28,0,0)
610 GOTO 400
620 CALL DELSPRITE(#A)
630 CALL SOUND(100,-7,0)
640 CALL MOTION(#28,0,0)
650 NEXT A
660 CALL CHARSET
670 PRINT "YOUR SCORE IS ":COUNT
680 PRINT "PRESS 'T' TO PLAY AGAIN:"
690 CALL KEY(0,GOT.STATUS)
700 IF STATUS=0 THEN 690
710 IF GOT=89 THEN 280
720 PRINT "BYE"
```

DOGFIGHT

EXTENDED BASIC



Dogfight is a two-player game written in Extended BASIC. Each player has control of one aircraft—a biplane. You must outmaneuver your opponent and shoot him down before he can do the same to you. If both planes crash into each other, the score will not change. The first player to destroy 10 enemy aircraft wins the game.

Your plane is controlled with four directional keys. Unlike most games, the key pressed will not cause your plane to immediately move in that particular direction. For example, if your plane is traveling down and to the right at a bearing of 135 degrees and you press E or the up arrow, your plane will turn its nose up and first change its heading to 90 degrees, then to 45 degrees, and finally to due north—i.e., straight up. This gives the plane a more realistic movement and makes unrealistic, 180-degree hairpin turns impossible.

To make a 180-degree turn to go due west, you must first press either E or X to indicate the upward or downward turn. Pressing S, the left directional key, will have no effect. In a similar way, the player using the right-hand side of the keyboard uses I, J, K, and M to move the plane. (If you have the overlay for the *Video Games* Command Cartridge, you might find it convenient.)

Pressing the F and H keys will fire the guns, but only one shot can be fired at a time. Each shot has a limited range and cannot be carried over the edge of the screen to the opposite edge. You can't terminate a bad shot; it must first go off screen. The limit of only one shot at a time was placed in the program so that the computer could make accurate coincidence checks with rapidly moving objects on the screen.

Four levels of difficulty make the game easy enough for beginners and challenging enough to hold the interest of experts. The higher the level of difficulty, the faster the planes and shots will move. The option to fly a day or night mission will change the screen color to either light blue or black.

Features of Extended BASIC Used in *Dogfight*

For you *99'er* readers who are also programmers, *Dogfight* illustrates many features of Extended BASIC.

Lines 190-280 define special graphics characters four at a time. Then line 290, CALL MAGNIFY(3) indicates that sprites will actually consist of four regular size characters, and only the first character number needs to be specified.

CALL SPRITE(#1,96,2,120,20,0,5) defined Sprite #1 (the first plane) as characters 96, 97, 98, and 99, black, starting in dot-row 120 and dot-column 20, and going zero velocity up or down, and to the right at velocity 5. More than one sprite may be defined at a time, as in line 310.

CALL DELSPRITE(#1) deletes Sprite #1, and more than one sprite may be deleted at a time. This statement is used when the planes are hit or they crash, or when the bullet leaves the edge of the screen.

Complex IF-THEN-ELSE statements are used in lines 530-780 to determine in what direction the plane is headed. CALL PATTERN then draws the plane depending on its heading; all other sprite characteristics remain the same.

CALL POSITION(#1,B1,B2) in line 1000 returns the row and column position of Sprite #1 so the bullet can be shot from that same position.

CALL MOTION can change the motion of a sprite without affecting other characteristics.

CALL COINC(#2,#3,3*V,PC) determines if Sprite #2 (second plane) and Sprite #3 (bullet from first plane) are coincident within a tolerance of 3*V. If so, a value of −1 is returned for PC. Coincidence is reported only when the CALL COINC statement is actually executed during the program. At greater velocities of the sprites, coincidence will not be detected if the program is busy elsewhere.

There are several ways to avoid concidences not being detected: 1) CALL COINC more often; 2) increase the tolerance level with increasing velocity; or 3) increase the execution speed so that the CALL COINC statements are executed more often. But, as with many programming problems, the solution involves a trade-off: The first slows down other action; the second could have planes crash when they're not touching; the third means cutting back on other types of action. One low-cost way to speed up execution, however, is to use multiple statements per line: CALL MOTION (#1,0,V)::RETURN executes faster than two separate lines with those statements.

Although kept to a minimum, the coincidence problem does still exist in this program: Once in a while a bullet will pass right through a plane. You'll just have to visualize a three-dimensional situation—a bullet passing directly over or under the plane but at a different altitude. Also, once in a while a bullet won't disappear at the edge of the screen, but will "wrap." Just consider it a stray bullet—a frequent occurrence in a real dogfight.

EXPLANATION OF THE PROGRAM

Dogfight

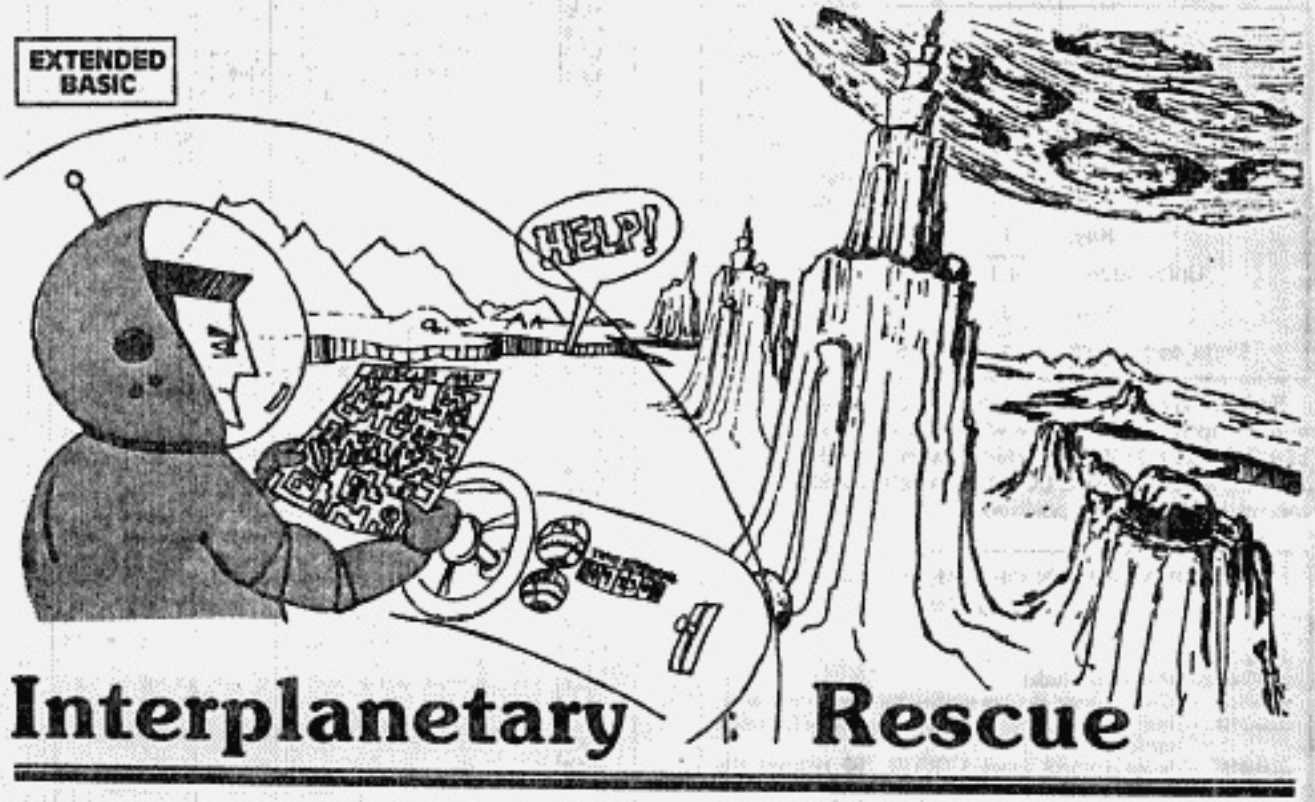
Line Nos.	
170	Clears screen, makes screen light blue.
180	P is the orientation of Plane 1; P1 the orientation of Plane 2.
190-290	Defines special characters for graphics. Using CALL MAGNIFY(3) allows sprites to be defined as 4 regular-sized graphics characters.
300-340	Draws title screen with two planes and sound.
350-360	Receives players' names for scoring.
370-420	Allows choice of one of four options.
430-460	Allows choice of day or night mission.
470-490	Starts *Dogfight* with planes going toward each other at velocity V.
500	Waits for players to press key.
510	If no key is pressed, branches to checking coincidence.
520-660	If key on left half of keyboard is pressed, checks which key and either moves Plane #1 that direction or shoots.
670-810	If key on right half of keyboard is pressed, moves Plane #2 appropriately or shoots.
820-850	Checks for coincidence between bullets and planes or between planes. If the sprites are coincident, branches to section of program for crashing. If not, continues sound of flying.
860-890	Deletes the bullet at the edges of the screen.
900	Returns to line 500 to check players' action.
910-980	Subroutines to move Plane #1 the correct direction after turning.
990	Returns if a bullet from Plane #1 is in motion.
1000-1100	Starts a bullet from the plane.
1110-1180	Subroutines to move Plane #2 the correct direction.
1190-1210	Returns if a bullet from Plane #2 is already in motion; otherwise shoots from Plane #2.
1220-1340	Procedure if planes crash into each other; makes crashing noise; draws planes falling; deletes sprites at the bottom of the screen.
1350-1410	Procedure if Plane #1 is shot.
1420-1490	Increments score for Player 2 and prints score.
1500-1560	Procedure if Plane #2 is shot.
1570-1630	Increments score for Player 1 and prints scores.
1640	Ends game if either player's score is 10.
1650-1680	If score is less than 10, players may choose to try again or stop.
1690-1720	Prints final score and ending remarks.

```
100 REM *************
110 REM * DOG FIGHT *
120 REM *************
130 REM
140 REM
150 REM
160 REM
170 CALL CLEAR :: CALL SCREEN(6)
180 P=1 :: P1=5
190 CALL CHAR(96,"00000000001C0E2F73F3E
071020000000000000000F8017DFDFF05FD
217070")
200 CALL CHAR(100,"000000102040901030706
FFFF7E4A0202209000086CFAE9D8897CDC9
8")
210 CALL CHAR(104,"0F010777FFFFFFFFFF7F0
007070300600E1EE000C0FCFEFEFEFEFCC0C
0C000C010F0")
220 CALL CHAR(108,"0400910365F9710D5E3
3100000000000000000040200000D0E060F
FFF7C524040")
230 CALL CHAR(112,"00000000001F003EBFFFA
03F840E0E000000000000008103 47EFFCFCF
00004")
240 CALL CHAR(116,"00000000000500112747 1
E0D4B3C1000804303C383CF8FF3960C080C
0E6C0")
250 CALL CHAR(120,"0F0703010303033F7F7
F7F7F3F0300077870 60C0E0E060FEFFFFF
FFFFEE0000F0")
260 CALL CHAR(124,"0C3C1C1C3C1FFF0D060
30107070300000000000000A81000E4F2783
9D27C081020")
270 CALL CHAR(128,"00000000000000000001010
0000000000000000000000000000000080")
280 CALL CHAR(132,"FC7E3C3C0150C0401203
0301C1F03000 0FC7E3870F0004000040C0
C38F0C0")
290 CALL MAGNIFY(3)
300 DISPLAY AT(12,9):"DOG FIGHT"
310 CALL SPRITE(#1,96,2,120,20,0,5,#2,
112,2,70,240,0,-5)
320 FOR TD=1 TO 2 :: CALL SOUND(1500,1
10,2,220,2,1000,30,-8,2)
330 NEXT TD :: CALL SOUND(1,9999,30)
340 CALL DELSPRITE(#1,#2):: CALL CLEAR
350 DISPLAY AT(5,1):"ENTER FIRST PLAYE
R'S NAME:" :: ACCEPT AT(7,3)SIZE(8
):PLA1$
360 DISPLAY AT(11,1):"ENTER SECOND PLA
YER'S NAME:" :: ACCEPT AT(13,3)SIZ
E(8):PLA2$
370 CALL CLEAR
380 DISPLAY AT(4,3):"OPTIONS:"
390 DISPLAY AT(6,5):"1. BEGINNER" :: D
ISPLAY AT(8,5):"2. NOVICE"
400 DISPLAY AT(10,5):"3. INTERMEDIATE"
 :: DISPLAY AT(12,5):"4. PROFESSIO
NAL"
410 CALL KEY(0,KEY,S)
420 IF KEY<49 OR KEY>52 THEN 410 ELSE
V=2*(KEY-48)
430 CALL CLEAR :: DISPLAY AT(5,3):"DO
YOU WANT A DAY MISSION, OR A NIGH
T MISSION? (D/N)"
440 CALL KEY(0,KEY,S)
450 IF KEY=68 THEN CO1=6 :: CO2=2 :: G
OTO 470
460 IF KEY=78 THEN CO1=2 :: CO2=6 ELSE
 440
470 CALL CLEAR
480 CALL SCREEN(CO1)
490 CALL SPRITE(#1,96,8,70,16,0,V,#2,1
12,11,70,240,0,-V)
500 CALL KEY(1,K1,S1):: CALL KEY(2,K2,
S2)
510 IF S1=0 AND S2=0 THEN 820
520 IF S1=0 THEN 680
530 IF K1<>5 THEN 570 ELSE IF P>5 AND
P<8 THEN P=P-1 ELSE IF P<3 OR P=8
THEN P=P+1
540 IF P=0 THEN P=8
550 IF P=9 THEN P=1
560 GOTO 650
570 IF K1<>0 THEN 610 ELSE IF P>5 AND
P<7 THEN P=P+1 ELSE IF P<3 OR P=8
THEN P=P-1
580 IF P=9 THEN P=1
590 IF P=0 THEN P=8
600 GOTO 650
610 IF K1<>2 THEN 630 ELSE IF P>5 THEN
 P=P-1 ELSE IF P<5 AND P>1 THEN P=
P+1
620 GOTO 650
630 IF K1<>3 THEN 660 ELSE IF P<5 AND
P>1 THEN P=P-1 ELSE IF P>5 THEN P=
P+1
640 IF P=9 THEN P=1
650 CALL PATTERN(#1,(P*4)+92):: ON P G
OSUB 910,920,930,940,950,960,970,9
80 :: GOTO 670
660 IF K1=12 THEN GOSUB 990
670 IF S2=0 THEN 820
680 IF K2<>5 THEN 720 ELSE IF P1>3 AND
 P1<8 THEN P1=P1-1 ELSE IF P1<3 OR
 P1=8 THEN P1=P1+1
```

```
690 IF P1=0 THEN P1=8
700 IF P1=9 THEN P1=1
710 GOTO 800
720 IF E2<>0 THEN 760 ELSE IF P1>5 AND
    P1<7 THEN P1=P1+1 ELSE IF P1<5 OR
    P1=8 THEN P1=P1-1
730 IF P1=0 THEN P1=8
740 IF P1=9 THEN P1=1
750 GOTO 800
760 IF E2<>2 THEN 780 ELSE IF P1>5 THE
    N P1=P1-1 ELSE IF P1<5 AND P1>1 TH
    EN P1=P1+1
770 GOTO 800
780 IF E2<>3 THEN 810 ELSE IF P1<5 AND
    P1>1 THEN P1=P1-1 ELSE IF P1>5 TH
    EN P1=P1+1
790 IF P1=9 THEN P1=1
800 CALL PATTERN(#2,(P1*4)+92)::  ON P1
    GOSUB 1110,1120,1130,1140,1150,11
    60,1170,1180 :: GOTO 820
810 IF E2=1 THEN GOSUB 1190
820 CALL COINC(#2,#3,3*V,PC)::  IF PC=-
    1 THEN 1500
830 CALL COINC(#1,#4,3*V,PC)::  IF PC=-
    1 THEN 1350
840 CALL COINC(#1,#2,2,5*V,PC)::  IF PC
    =-1 THEN 1220
850 CALL SOUND(-4250,110,2,220,2,-8,2)
860 IF SH1=0 THEN 880
870 CALL POSITION(#3,B3,B4)::  IF B3<6
    OR B3>186 OR B4<8 OR B4>250 THEN C
    ALL DELSPRITE(#3)::  SH1=0
880 IF SH2=0 THEN 900
890 CALL POSITION(#4,C3,C4)::  IF C3<6
    OR C3>186 OR C4<8 OR C4>250 THEN C
    ALL DELSPRITE(#4)::  SH2=0
900 GOTO 500
910 CALL MOTION(#1,0,V)::  RETURN
920 CALL MOTION(#1,-V*.6,V*.8)::  RETURN
930 CALL MOTION(#1,-V,0)::  RETURN
940 CALL MOTION(#1,-V*.6,-V*.6)::  RETU
    RN
950 CALL MOTION(#1,0,-V)::  RETURN
960 CALL MOTION(#1,V*.6,-V*.6)::  RETURN
970 CALL MOTION(#1,V,0)::  RETURN
980 CALL MOTION(#1,V*.6,V*.6)::  RETURN
990 IF SH1=1 THEN RETURN
1000 CALL POSITION(#1,B1,B2)
1010 P3=P ::  SP=3 ::  A1=B1 ::  A2=B2 ::
    SH1=1
1020 ON P3 GOTO 1030,1040,1050,1060,107
    0,1080,1090,1100
1030 CALL SPRITE(#SP,128,CO2,A1,A2,0,V*
    2)::  RETURN
1040 CALL SPRITE(#SP,128,CO2,A1,A2,-V*1
    .6,V*1.6)::  RETURN
1050 CALL SPRITE(#SP,128,CO2,A1,A2,-V*2
    ,0)::  RETURN
1060 CALL SPRITE(#SP,128,CO2,A1,A2,-V*1
    .6,-V*1.6)::  RETURN
1070 CALL SPRITE(#SP,128,CO2,A1,A2,0,-V
    *2)::  RETURN
1080 CALL SPRITE(#SP,128,CO2,A1,A2,V*1.
    6,-V*1.6)::  RETURN
1090 CALL SPRITE(#SP,128,CO2,A1,A2,V*2,
    0)::  RETURN
1100 CALL SPRITE(#SP,128,CO2,A1,A2,V*1.
    6,V*1.6)::  RETURN
1110 CALL MOTION(#2,0,V)::  RETURN
1120 CALL MOTION(#2,-V*.6,V*.8)::  RETURN
1130 CALL MOTION(#2,-V,0)::  RETURN
1140 CALL MOTION(#2,-V*.6,-V*.6)::  RETU
    RN
1150 CALL MOTION(#2,0,-V)::  RETURN
1160 CALL MOTION(#2,V*.6,-V*.6)::  RETURN
1170 CALL MOTION(#2,V,0)::  RETURN
1180 CALL MOTION(#2,V*.6,V*.6)::  RETURN
1190 IF SH2=1 THEN RETURN
```

```
1200 CALL POSITION(#2,C1,C2)
1210 A1=C1 ::  A2=C2 ::  P3=P1 ::  SP=4 ::
    SH2=1 ::  GOTO 1020
1220 CALL MOTION(#1,0,0,#2,0,0)::  CALL
    POSITION(#1,P01,P02,#2,P03,P04)
1230 CALL SPRITE(#1,132,16,P01,P02,20,0
    ,#2,132,16,P03,P04,20,0)
1240 CALL POSITION(#1,CR1,CR2)
1250 CALL POSITION(#2,CR3,CR4)
1260 IF CR1>186 OR CR3>186 THEN CALL DE
    LSPRITE(#1,#2)
1270 CALL SOUND(-1000,110,2,220,2,-7,0)
1280 IF CR1<>0 OR CR3<>0 THEN 1240
1290 CALL DELSPRITE(#1,#2,#3,#4)
1300 CALL SCREEN(6)
1310 DISPLAY AT(5,15)::"CRAAAASSH"
1320 DISPLAY AT(10,5)::"BOTH TEAMS CRASH
    ED."
1330 DISPLAY AT(12,5)::"NO POINTS AWARDE
    D."
1340 GOTO 1640
1350 CALL DELSPRITE(#3,#4)
1360 CALL MOTION(#1,0,0)::  CALL POSITIO
    N(#1,P01,P02)
1370 CALL SPRITE(#1,132,16,P01,P02,20,0
    )
1380 CALL SOUND(-1000,440,2,220,2,-4,0)
1390 CALL POSITION(#1,CR1,CR2)
1400 IF CR1<186 THEN 1380
1410 CALL DELSPRITE(#1,#2)
1420 RED=RED+1
1430 CALL SCREEN(6)
1440 DISPLAY AT(5,1)::PLA2$;"  HAS WON TH
    E BATTLE."
1450 DISPLAY AT(8,3)::PLA2$;"  SCORE IS  :
    "
1460 DISPLAY AT(9,26)::RED
1470 DISPLAY AT(11,3)::PLA1$;"  SCORE IS
    :"
1480 DISPLAY AT(11,26)::BLUE
1490 GOTO 1640
1500 CALL DELSPRITE(#3,#4)
1510 CALL MOTION(#2,0,0)::  CALL POSITIO
    N(#2,P01,P02)
1520 CALL SPRITE(#2,132,16,P01,P02,20,0
    )
1530 CALL SOUND(-1000,440,2,220,2,-4,0)
1540 CALL POSITION(#2,CR3,CR4)
1550 IF CR3<186 THEN 1530
1560 CALL DELSPRITE(#2,#1)
1570 BLUE=BLUE+1
1580 CALL SCREEN(6)
1590 DISPLAY AT(5,1)::PLA1$;"  HAS WON TH
    E BATTLE."
1600 DISPLAY AT(8,3)::PLA1$;"  SCORE IS  :
    "
1610 DISPLAY AT(8,26)::BLUE
1620 DISPLAY AT(10,3)::PLA2$;"  SCORE IS
    :"
1630 DISPLAY AT(10,26)::RED
1640 IF RED=10 OR BLUE=10 THEN 1690
1650 DISPLAY AT(20,3)::"WANT TO TRY AGAI
    N? (Y/N)."
1660 PC=0 ::  SH1=0 ::  SH2=0 ::  P=1 ::  P
    1$
1670 CALL KEY(0,KEY,S)
1680 IF KEY=89 THEN 570 ELSE IF KEY<>78
    THEN 1670
1690 IF BLUE>RED THEN PRINT PLA1$;"  HAS
    WON THE WAR.":"SORRY, ";PLA2$ ::
    GOTO 1720
1700 IF RED>BLUE THEN PRINT PLA2$;"  HAS
    WON THE WAR.":"SORRY, ";PLA1$ ::
    GOTO 1720
1710 PRINT "THE WAR HAS ENDED IN A TIE.
    BOTH SIDES ARE NOW AT PEACE."
1720 END
```

EXTENDED BASIC



Interplanetary Rescue

You are sitting there enjoying a cup of coffee at the Interplanetary Rescue Lounge when the news arrives: A cave-in on Moon Base 2 has seriously injured a miner. Instantly you race for the shuttle, knowing you must reach the moon base, pick up the injured miner, and return to the base medical facilities. There's no time to waste!

Your ship is fueled and ready at the docking pad. In your ship you sit in front of your TI-99/4A controller panel and view the radar and instrumentation screen. You are now ready to take off. Press T on the control panel and the shuttle begins to lift. Using your six thrust control buttons, you adjust your climb to the desired level. The terrain between you and Moon Base 2 is treacherous, and you must quickly ascertain the best route. Using your horizontal thrusters (arrow keys), you start your trip across the lunar landscape.

Accidents may happen anywhere, and right now your Interplanetary Rescue Team is in charge of the moon, Mars, and Venus. Use the arrow keys (E,S,D,X) to control horizontal movement. Horizontal velocity is listed on your control screen.

The elevations of the terrain show up as different colors on the map. At the right-hand side of the screen is a visual representation of your altitude in relation to the elevation colors. Your ship must be above the color on the right of the screen to safely pass through that color on the radar screen. Be careful not to overshoot the highest elevation color you plan to cross: You will waste valuable time getting back down and precious fuel needed for the return voyage.

When you land on Moon Base 2, you must be traveling at a vertical speed of less than 6 meters/second to make a perfect landing. A rough landing of 6-10 meters/second will cause a leak in your main fuel valve, causing a loss of half your fuel. Any faster than 10 meters/second and you'll crash, never to return home. Once safely on the ground, the injured miner is put on board, and you're ready for the return trip. You won't have as much fuel holding you down, so it won't take as much thrust to accelerate vertically. Good luck on your rescue mission . . . you may need it!

Instrument Displays:

ALT = altitude in meters. Each succeeding color on the radar represents 2000 meters.

HVEL = horizontal velocity across the radar screen (dependent upon arrow keys).

VVEL = vertical velocity; + is climbing, – is falling.

TIME = number of seconds into the mission.

FUEL = weight of fuel remaining, in kilograms.

PWR = amount of thrust being generated. Each unit of thrust equals 1000 newtons; each newton equals approximately 2.05 kilograms of thrust.

Calculations and Variables

The gravity formula in line 950 is the formula for speed of a falling object. V2 is the change in velocity in m/sec, F is the thrust in newtons, S is the weight of the ship in kg., E is the weight of remaining fuel in kg., and G is the gravity in m/sec^2. The time is one second in this calculation. All variables starting with D pertain to distance, and H is the altitude.

Graphics

Characters accessible on the keyboard but not used in printing messages have been redefined (lines 190-290). Then by using DISPLAY AT and a string, you can display colorful graphics very quickly without calling each square one-by-one. This method was used to display the radar map

much more quickly than by using HCHAR and VCHAR. The strings are read in as 21 DATA statements. By changing the DATA statements in this program or adding some of your own, you may easily change the maps.

Option chosen	G (Gravity)	E (Fuel)	Take-off thrust
Moon	2	20000	65000
Mars	4	45000	230000
Venus	6	80000	540000

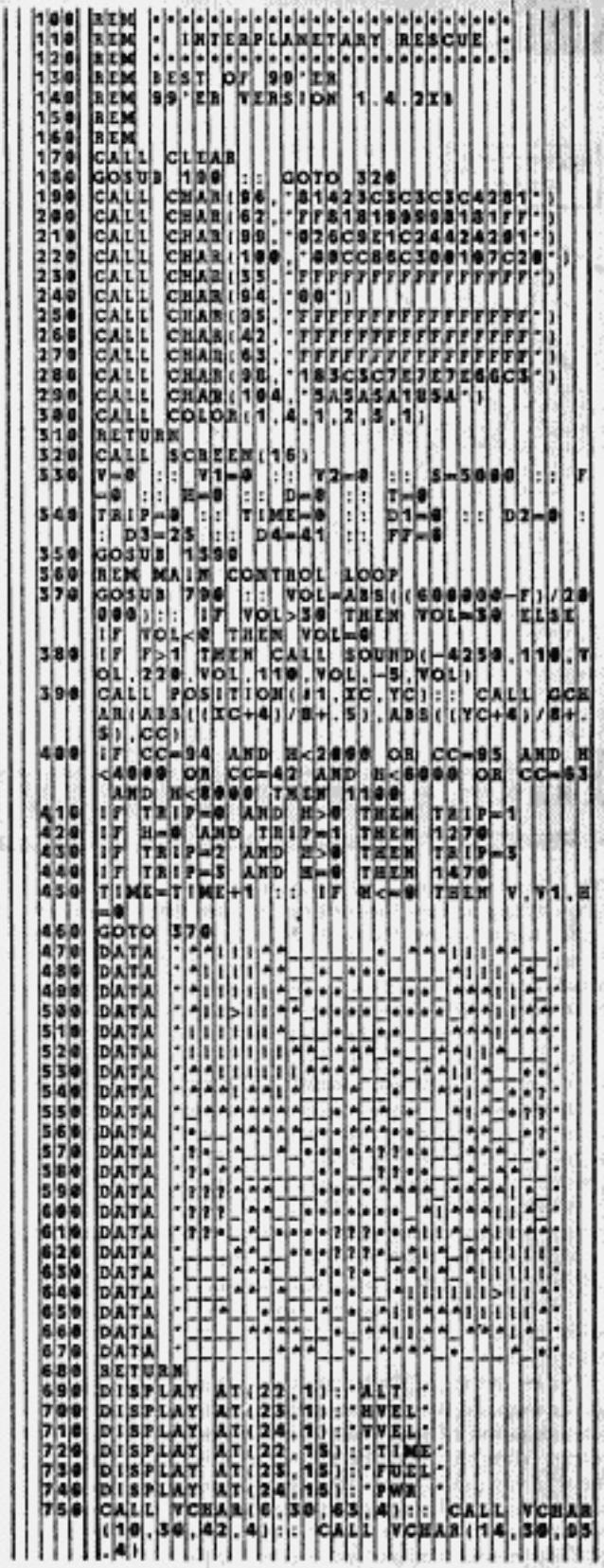
Thrust keys: T = initial thrust (displayed as 65, 230, or 540).

Key:	I	O	P
Units added:	+1	+5	+10
Key:	J	K	L
Units decreased:	−1	−5	−10

Sprites were used to depict the two crafts on the screen in order to be able to move with high resolution. Instead of using CALL MOTION which makes control of position difficult, we used CALL LOCATE which provides absolute control of the sprite's position.

EXPLANATION OF THE PROGRAM
Interplanetary Rescue

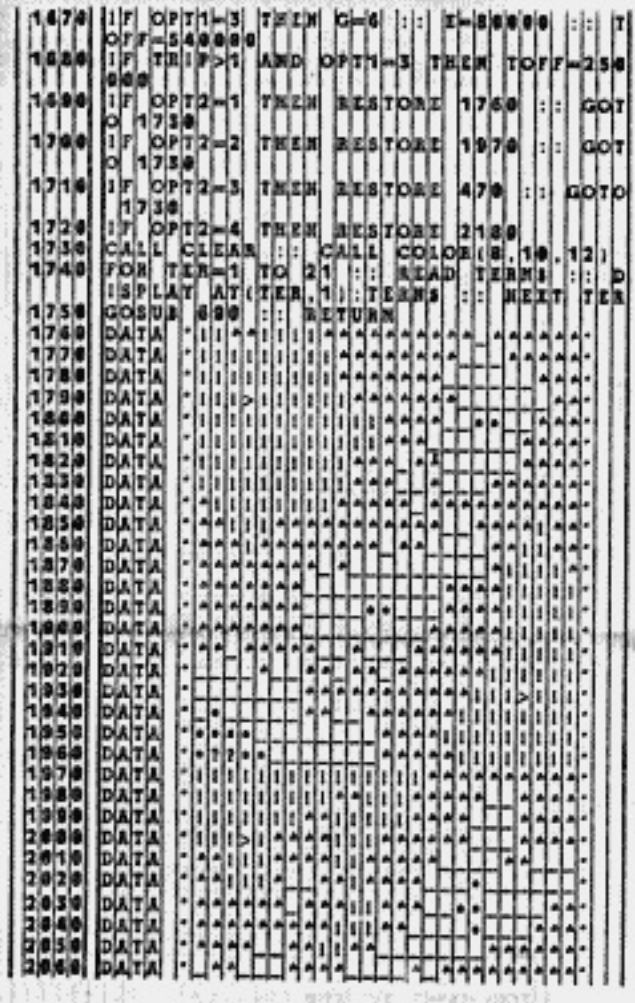
Line Nos.	
100-160	Program header
170-310	Clears screen; defines special characters and colors.
320-350	Initializes variables; branches to title screen and options.
360-460	Main control loop; GOSUB 790 receives the player's key presses. The VOL in the CALL SOUND statement depends upon the power. H is the altitude, and line 400 tests for crashes. If the rescue craft has landed at either base, the program branches.
470-680	DATA statements to draw the "Novice" map.
690-780	Subroutine to label the parameters and draw the altimeter.
790-1020	Receives player's key responses and calculates parameters.
1030-1090	Prints updated altitude, time, velocities, fuel, and power.
1100-1150	Message and procedure for crashing into the hill.
1160-1180	Subroutine for procedure for any crash.
1190-1260	Prints score and option to play again; branches appropriately.
1270-1400	Messages and procedure for crashing at high velocity.
1410-1460	Procedure if craft lands safely; starts return trip.
1470-1580	Procedure for craft landing on return trip.
1590-1640	Prints and receives options of planet and level of difficulty.
1650-1750	Depending on the options chosen, sets gravity, fuel, and initial thrust, then prints appropriate map.
1760-2380	DATA statements for three maps.
2390-2460	Prints title screen.

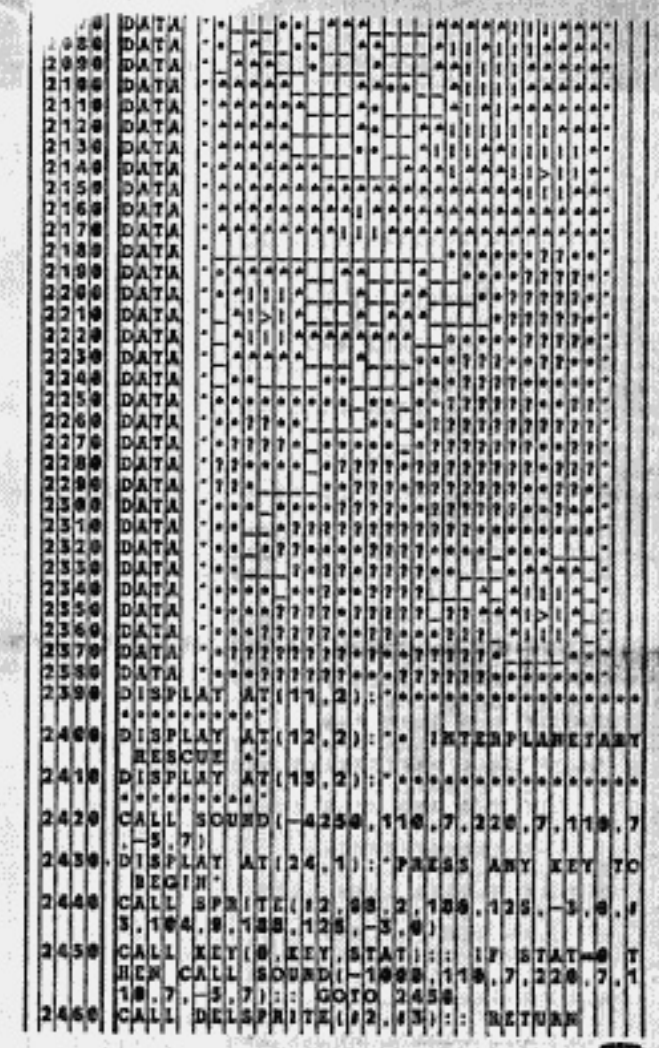


```
760 CALL VCHAR(18,30,94,4):: CALL HCHA
R(22,28,33,3)
770 CALL SPRITE(#2,98,2,160,222,#1,96,
2,D3,D4)
780 RETURN
790 CALL KEY(1,K1,S1):: CALL KEY(2,K2,
S2):: IF S1=0 AND S2=0 THEN 920
800 IF S1=0 THEN 860
810 IF K1=5 THEN D1=D1-1 :: E=E-50 ::
GOTO 860
820 IF K1=0 THEN D1=D1+1 :: E=E-50 ::
GOTO 860
830 IF K1=2 THEN D2=D2-1 :: E=E-50 ::
GOTO 860
840 IF K1=3 THEN D2=D2+1 :: E=E-50 ::
GOTO 860
850 IF K1=11 THEN F=TOFF
860 IF K2=3 THEN F=F-5000 :: GOTO 920
870 IF K2=12 THEN F=F-10000 :: GOTO 92
0
880 IF K2=2 THEN F=F-1000 :: GOTO 920
890 IF K2=5 THEN F=F+1000 :: GOTO 920
900 IF K2=6 THEN F=F+5000 :: GOTO 920
910 IF K2=11 THEN F=F+10000
920 IF E<=0 THEN E=0 :: F=0
930 IF F<0 THEN F=0
940 IF F=0 THEN CALL PATTERN(#3,32)ELS
E CALL PATTERN(#3,104)
950 V2=F/(S+E)-G :: V=V+V2 :: DV=V ::
 IF V<0 AND H<=0 THEN DV=0
960 D=(V1+(V2/2)):: V1=V :: H=H+D :: E
=E-(ABS(F/2000))
970 IF H<=0 THEN H=0
980 IF H>9935 THEN 1090
990 CALL LOCATE(#2,160-(H/(500/8)),222
,#3,168-(H/(500/8)),222)
1000 D3=D3+D1 :: IF D3<1 THEN D3=1 ELSE
 IF D3>160 THEN D3=160
1010 D4=D4+D2 :: IF D4<17 THEN D4=17 EL
SE IF D4>208 THEN D4=208
1020 CALL LOCATE(#1,D3,D4)
1030 DISPLAY AT(22,5)SIZE(6):H
1040 DISPLAY AT(22,20)SIZE(5):TIME
1050 DISPLAY AT(23,5)SIZE(5):SQR(D1^2+D
2^2)*62.5
1060 DISPLAY AT(23,20)SIZE(6):E
1070 DISPLAY AT(24,5)SIZE(5):DV
1080 DISPLAY AT(24,20)SIZE(7):F/1000
1090 RETURN
1100 CALL HCHAR(22,1,32,96):: GOSUB 116
0
1110 CALL CHARSET :: DISPLAY AT(22,3):"
YOU CRASHEDD INTO THE HILL."
1120 IF TRIP=1 AND TIME<250 THEN CR=TIM
E/2000 ELSE CR=.15
1130 DISPLAY AT(23,1):" ALTITUDE=";H
1140 DISPLAY AT(24,3):"VELOCITY=";V
1150 GOTO 1190
1160 FOR REP=1 TO 5 :: CALL SOUND(300,1
10,0,110,0,110,0,-8,0):: CALL PATT
ERN(#1,99)
1170 CALL SOUND(600,110,0,110,0,220,0,-
4,0):: CALL PATTERN(#1,100):: NEXT
 REP
1180 CALL CLEAR :: CALL DELSPRITE(ALL):
: CALL CHARSET :: RETURN
1190 FOR TD=1 TO 500
[illegible]
1240 IF ANS$="N" THEN 1260 ELSE IF ANS$
<>"Y" THEN 1230
1250 CALL CLEAR :: GOTO 170
1260 STOP
1270 CALL HCHAR(22,1,32,96)
1280 IF XC<>137 AND YC<>185 THEN GOSUB
 1160 :: CR=.08 :: DISPLAY AT(22,1
):"YOU MISSED THE PAD.":"YOUR SHIP
HAS CRASHED." :: GOTO 1190
1290 IF V>-30 THEN 1330 ELSE GOSUB 116
 :: DISPLAY AT(22,1):"YOU BLEW IT.
 YOU LEFT A"
1300 DISPLAY AT(23,1):"CRATER A MILE W
DE." :: CR=.1
1310 DISPLAY AT(24,1):"VELOCITY=";V
1320 GOTO 1190
1330 IF V>-20 THEN 1370 ELSE GOSUB 116
 :: DISPLAY AT(22,1):"A BAD LANDI
G-TWO CREWMEN"
1340 DISPLAY AT(23,1):"ARE DEAD, AND YO
U ARE HURT." :: CR=.15
1350 DISPLAY AT(24,1):"VELOCITY=";V
1360 GOTO 1190
1370 IF V>-10 THEN 1410 ELSE GOSUB 116
 :: DISPLAY AT(22,1):"YOUR SHIP IS
 BADLY DAMAGED."
1380 DISPLAY AT(23,1):"THIS IS YOUR LAS
T FLIGHT." :: CR=.25
1390 DISPLAY AT(24,1):"VELOCITY=";V
1400 FF=FF-3000 :: GOTO 1190
1410 IF V>-6 THEN 1430 ELSE CALL CLEAR
 :: CALL CHARSET :: DISPLAY AT(22,1
):"A ROUGH LANDING. YOU HAVE"
1420 DISPLAY AT(23,1):"LOST 1/2 OF YOUR
 FUEL." :: E=E/2 :: CR=.7 :: GOTO
1450
1430 CALL CLEAR :: CALL CHARSET :: DISP
LAY AT(22,1):"A PERFECT LANDING. Y
OU ARE"
1440 DISPLAY AT(23,1):"IN GOOD SHAPE TO
 RETURN." :: CR=1
1450 DISPLAY AT(24,1):"VELOCITY=";V ::
FOR TD=1 TO 500
1460 NEXT TD :: F=0 :: TRIP=2 :: GOSUB
 190 :: GOSUB 1680 :: GOTO 370
1470 CALL DELSPRITE(#1,#2,#3)
1480 CALL CLEAR
1490 CALL CHARSET :: IF V<=-10 THEN 127
0 ELSE CALL HCHAR(22,1,32,96)
1500 IF V>-6 THEN 1550
1510 IF TRIP=1 AND(XC<>137 OR YC<>185)T
HEN 1270
1520 IF TRIP=3 AND(D3<>25 OR D4<>41)THE
N 1270
1530 DISPLAY AT(22,1):"A ROUGH LANDING.
 YOUR SHIP BARELY MADE IT."
1540 CR=CR*.75 :: GOTO 1570
1550 DISPLAY AT(22,1):"CONGRADULATIONS,
 A PERFECT"
1560 DISPLAY AT(23,1):"LANDING. EARTH I
S PROUD."
1570 DISPLAY AT(24,1):"VELOCITY=";V ::
FOR TD=1 TO 2000
1580 NEXT TD :: GOTO 1190
1590 CALL CLEAR :: GOSUB 2300 :: CALL C
LEAR :: DISPLAY AT(1,7):"PLANET OF
TIONS"
1600 DISPLAY AT(3,1):"1. MOON" :: "2. MA
RS" :: "3. VENUS"
1610 ACCEPT AT(10,1)VALIDATE("123")SIZE
(1):OPT1 :: CALL CLEAR
[illegible]
E(1):OPT2
1650 IF OPT1=1 THEN G=2 :: E=20000 :: T
OFF=65000 :: GOTO 1680
1660 IF OPT1=2 THEN G=4 :: E=45000 :: T
OFF=230000 :: GOTO 1680
```

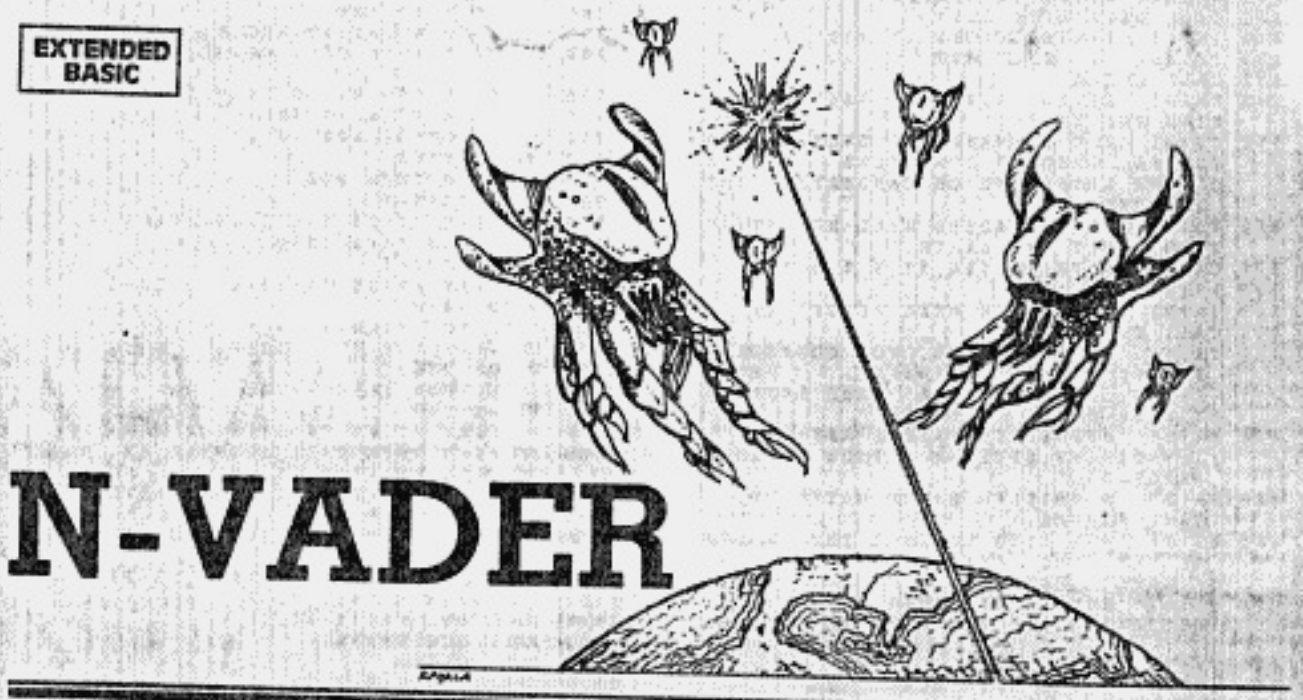
```
1200 next TD
1210 CALL CLEAR:: DISPLAY AT(20,3):" Wish to play again (Y/N) "
1220                 "   (10,5):" Your score is :"; INT((12000-2*TIME)+E/6+(OPT1*OPT2*500))ACCR)
  30                 "   (23,3)BEEP:ANS$
  40

1620                 "   (1,1):"Level of difficulty"
  30                 "   (3,1):" 1. sp. Beginner "::" 2. sp. intermediate ":: novice professional
  40 ACCEPT AT(12,1)VALIDATE("1234")SIZE(1):OPT2
```





EXTENDED BASIC



N-VADER

N-Vader is a game for one or two players written in Extended BASIC. Each player controls a "defense ship" whose mission is to prevent alien creatures from reaching Earth. The game is played using either the keyboard or joysticks. If joysticks are used with the TI-99/4A, be sure to put the ALPHA LOCK key in the *up* position *after* setting the parameters of game play in response to the screen prompts.

Aliens are destroyed by positioning the defense ship in the immediate vicinity of the alien. No fire button or key is needed. Every time an alien is destroyed, the player scores a point and another alien is introduced at the top of the screen. Whenever an alien reaches Earth (bottom of the screen), the aliens score.

One unusual feature of this game is its flexibility. When the game starts, the player(s) can select the number of aliens, their speed, the speed of the defense ship and the defense range. Defense range defines the proximity necessary for a defense ship to destroy an alien.

Features of Extended BASIC Used in *N-Vader*

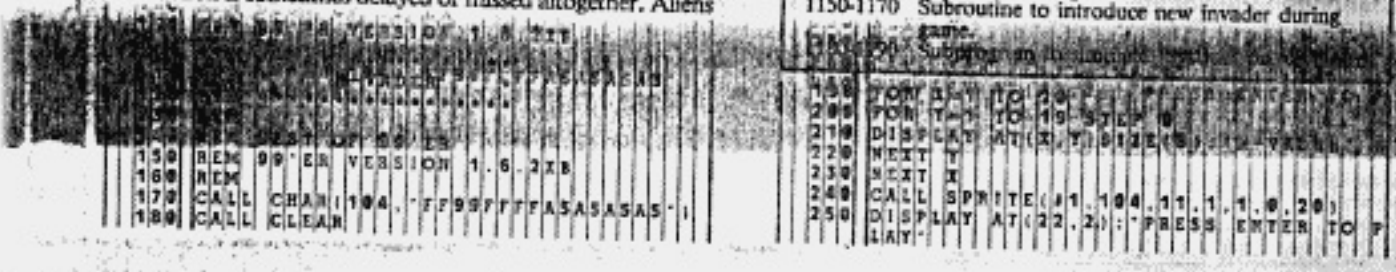
Several Extended BASIC features are used to make *N-Vader* work. Sprites are, of course, fundamental to the program. CALL DISTANCE is used to determine the proximity of alien and defensive ship(s). CALL COINC is used to determine when aliens reach Earth.

Because sprites move independently of the program, alien destruction is sometimes delayed or missed altogether. Aliens can also descend through the Earth for the same reason. Fortunately, these quirks of sprites actually make the game more enjoyable. For example, it is sometimes possible for a defense ship to swoop down *into* the Earth and pick off an alien at the last possible instant!

A subprogram (lines 1390-1510) is used to allow keyboard input to be processed by the main program as joystick input. Programmers with diskettes may want to save this subprogram in a MERGE file for inclusion in other programs.

EXPLANATION OF THE PROGRAM
N-Vader

Line Nos.	
100-160	Program header.
170	Define invader character.
180-270	Display title screen with sprites.
280-470	Instructions.
480-700	Get parameters.
710-790	Draw Earth.
800-820	Draw invader sprites.
830-840	Draw player sprites. Note that positioning changes for Player #1 depending upon number of players.
850-890	Check for player scoring.
900	Check for invaders at Earth.
910-940	Adjust player motion.
950-1140	Process end of game.
1150-1170	Subroutine to introduce new invader during game.
1390-1510	Subprogram to [illegible]



```
260 CALL SPRITE(#2,184,13,45,258,0,-20
    )
270 CALL SPRITE(#3,184,9,73,1,0,100)
280 CALL SPRITE(#4,184,4,81,256,0,-80)
290 ACCEPT AT(22,22)SIZE(-1):X$
300 CALL DELSPRITE(ALL)
310 INPUT "INSTRUCTIONS (Y/N)?":X$
320 IF X$<>"Y" THEN 490
330 CALL CLEAR
340 PRINT "ALIEN CREATURES ARE":"ATTAC
    KING THE EARTH!"
350 PRINT "YOU COMMAND THE ONLY DEFENS
    E SHIPS. ONBOARD COMPUTERS CONTR
    OL THE LASERS WHICH CANDESTROY THE
     INVADERS."
360 PRINT "THE INVADERS WILL NOT ATTAC
    K YOU. ONLY THE EARTH."
370 CALL SPRITE(#1,184,11,1,1,0,20)
380 PRINT
390 INPUT "HIT ENTER WHEN READY":X$
400 CALL CLEAR
410 PRINT "IN THIS GAME, YOU CONTROL
    THE NUMBER OF INVADERS, THEIR
    SPEED, YOUR SPEED & THE LASER R
    ANGE OF YOUR SHIPS."
420 PRINT "SUGGESTED VALUES ARE:
    INVADERS=6,SPEED=8":"YOUR SPEED=3
    ,RANGE=25."
430 PRINT "A SMALLER RANGE MAKES THE
    GAME HARDER."
440 PRINT "YOU ALSO CONTROL THE LENGTH
    OF THE GAME. WHEN ASKED 'END
    OF GAME', ENTER THE"
450 PRINT "NUMBER OF TIMES THE ALIENS
    HIT EARTH FOR THE GAME TO BEOVER.
    "
460 PRINT
470 INPUT "ENTER WHEN READY":X$
480 CALL DELSPRITE(#1)
490 HIT,ZAP1,ZAP2=0
500 INPUT "NUMBER OF PLAYERS? ":NP
510 IF NP<0 OR NP>2 THEN 500
520 NP=INT(NP)
530 INPUT "PLAYER 1 NAME? ":P1$
540 IF NP=1 THEN 560
550 INPUT "PLAYER 2 NAME? ":P2$
560 PRINT "NUMBER OF INVADERS?"
570 INPUT INV
580 IF INV<1 OR INV>8 THEN 560
590 PRINT "INVADER SPEED?"
600 INPUT IS
610 IF IS<1 THEN 590
620 PRINT "DEFENDER SPEED(1-9)?"
630 INPUT SPD
640 IF SPD<=0 THEN 620
650 PRINT "DEFENSE RANGE?"
660 INPUT RNG
670 IF RNG<1 OR RNG>200 THEN 650
680 PRINT "END OF GAME?"
690 INPUT WIN
700 INPUT "JOYSTICKS (Y/N)? ":X$
710 IF SEG$(X$,1,1)="Y" THEN JS=1
720 IF WIN<1 THEN 680
730 CALL CHAR(100,"FFFFFFFFFFFFFFFF")
740 CALL CHAR(96,"0003031C7F1C0808")
750 CALL SCREEN(2)
760 CALL CLEAR
770 CALL COLOR(9,16,16)
780 CALL COLOR(3,2,5)
790 CALL COLOR(4,2,3)
800 FOR X=22 TO 24
810 CALL HCHAR(X,1,100,32)
820 NEXT X
830 FOR X=1 TO INV
840 CALL SPRITE(#X,104,3+X,1,INT(RND*2
    56)+1,INT(RND*IS)+1,INT(RND*IS)-IS
    /2)
850 NEXT X
860 IF NP=1 THEN CALL SPRITE(#9,90,15,
    100,128)ELSE CALL SPRITE(#9,90,15,
    100,56)
870 IF NP=2 THEN CALL SPRITE(#10,90,15
    ,100,200)
880 FOR X=1 TO INV
890 CALL COINC(#X,#9,RNG,3)
900 IF NP=2 THEN CALL COINC(#X,#10,RNG
    ,32)
910 IF B>=0 AND B2>=0 THEN 1020
920 CALL PATTERN(#X,100)
930 CALL SOUND(-500,-3,0)
940 GOSUB 1360
950 IF B>=0 THEN 980
960 ZAP1=ZAP1+1
970 DISPLAY AT(23,3)SIZE(4):ZAP1
980 IF B2>=0 THEN 1060
990 ZAP2=ZAP2+1
1000 DISPLAY AT(23,25)SIZE(4):ZAP2
1010 GOTO 1080
1020 CALL POSITION(#X,Y(X),H(X))
1030 IF Y(X)<138 THEN 1080
1040 GOSUB 1360
1050 CALL SOUND(-50,-2,0)
1060 HIT=HIT+1
1070 DISPLAY AT(23,14)SIZE(4):HIT
1080 IF JS=1 THEN CALL JOYST(1,JX,JY)EL
    SE CALL KEYST(1,JX,JY)
1090 CALL MOTION(#9,-JY*SPD,JX*SPD)
1100 IF NP=1 THEN 1130
1110 IF JS=1 THEN CALL JOYST(2,JX,JY)EL
    SE CALL KEYST(2,JX,JY)
1120 CALL MOTION(#10,-JY*SPD,JX*SPD)
1130 IF HIT<WIN THEN 1360
1140 CALL DELSPRITE(ALL)
1150 CALL SCREEN(16)
1160 CALL CLEAR
1170 CALL COLOR(3,2,1)
1180 CALL COLOR(4,2,1)
1190 CALL SPRITE(#1,104,7,1,1,0,25)
1200 PRINT "GAME OVER"
1210 PRINT "EARTH UNITS";TAB(18);HIT
1220 PRINT P1$;" DESTOYED";ZAP1;" ALIEN
    S"
1230 PRINT TAB(10);INT(100*ZAP1/(HIT+ZA
    P1+ZAP2));" PER CENT"
1240 IF NP=2 THEN PRINT P2$;" DESTROYED
    ";ZAP2;" ALIENS"
1250 IF NP=2 THEN PRINT TAB(10);INT(100
    *ZAP2/(HIT+ZAP1+ZAP2));" PER CENT"
1260 FOR X=1 TO 100
1270 CALL SOUND(50,440,0)
1280 CALL SOUND(99,880,0)
1290 NEXT X
1300 CALL DELSPRITE(#1)
1310 CALL SCREEN(8)
1320 JS=0
1330 GOTO 170
1340 NEXT X
1350 GOTO 880
1360 CALL DELSPRITE(#X)
1370 CALL SPRITE(#X,104,15-X,1,INT(RND*
    256)+1,INT(RND*IS)+1,INT(RND*IS)-I
    S/2)
1380 RETURN
1390 SUB KEYST(N,X,Y)
1400 CALL KEY(N,Z,S)
1410 IF S=0 THEN X,Y=0 :: SUBEXIT
1420 IF Z=2 THEN X=-4 :: Y=0
1430 IF Z=4 THEN X=4 :: Y=0
1440 IF Z=5 THEN X=0 :: Y=4
1450 IF Z=6 THEN X=4 :: Y=4
1460 IF Z=12 THEN X,Y=4
1470 IF Z=5 THEN X=4 :: Y=0
1480 IF Z=14 THEN X=4 :: Y=-4
1490 IF Z=8 THEN X=0 :: Y=-4
1500 IF Z=15 THEN X=-4 :: Y=-4
1510 SUBEND
```

EXTENDED BASIC

SPACE PATROL

The Earth is at war! Another planet is trying to gain control of our solar system. You are the captain of a patrol ship armed with high-powered lasers. Your mission—destroy a fleet of 15 enemy supply ships en route to their Battle Star. But be careful, because the supply ships are armed with "killer satellites." When launched, the satellites will move in on your ship and self-destruct unless you destroy them first.

Your ship has a supply of 400 energy units, and energy is depleted by 10 units each time you fire your lasers. You also have a deflector shield that is automatically activated when a "killer" gets past your lasers and explodes near you. This will deplete your energy by 50 units. Your on-board computer will warn you if a "killer" has been launched.

At the start of the game, your gun sight will appear in the center of the screen. You may use a joystick or the arrow keys to position this on your target (depending on the option chosen at the start of the game). Then press either the FIRE button or the Y key to fire your lasers.

GOOD LUCK AND GOOD SHOOTING, CAPTAIN!!

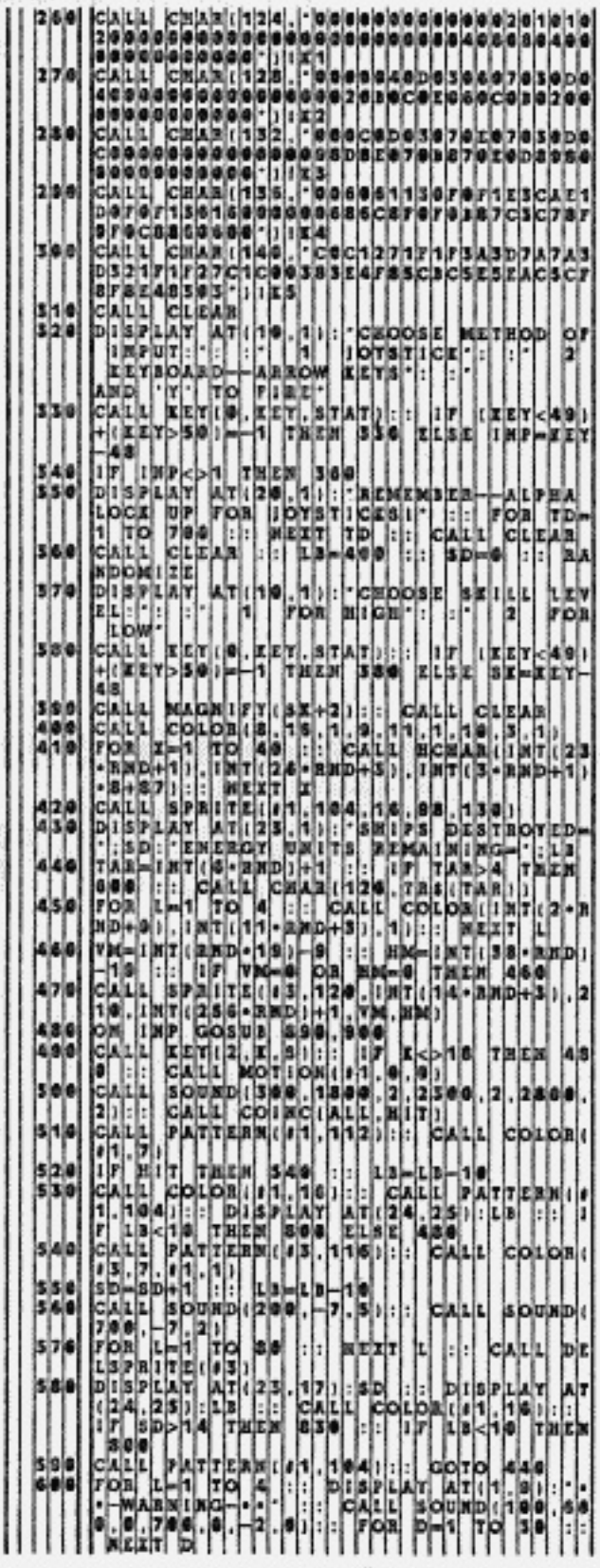
Note: If using joysticks with the TI-99/4A, release the ALPHA LOCK key.

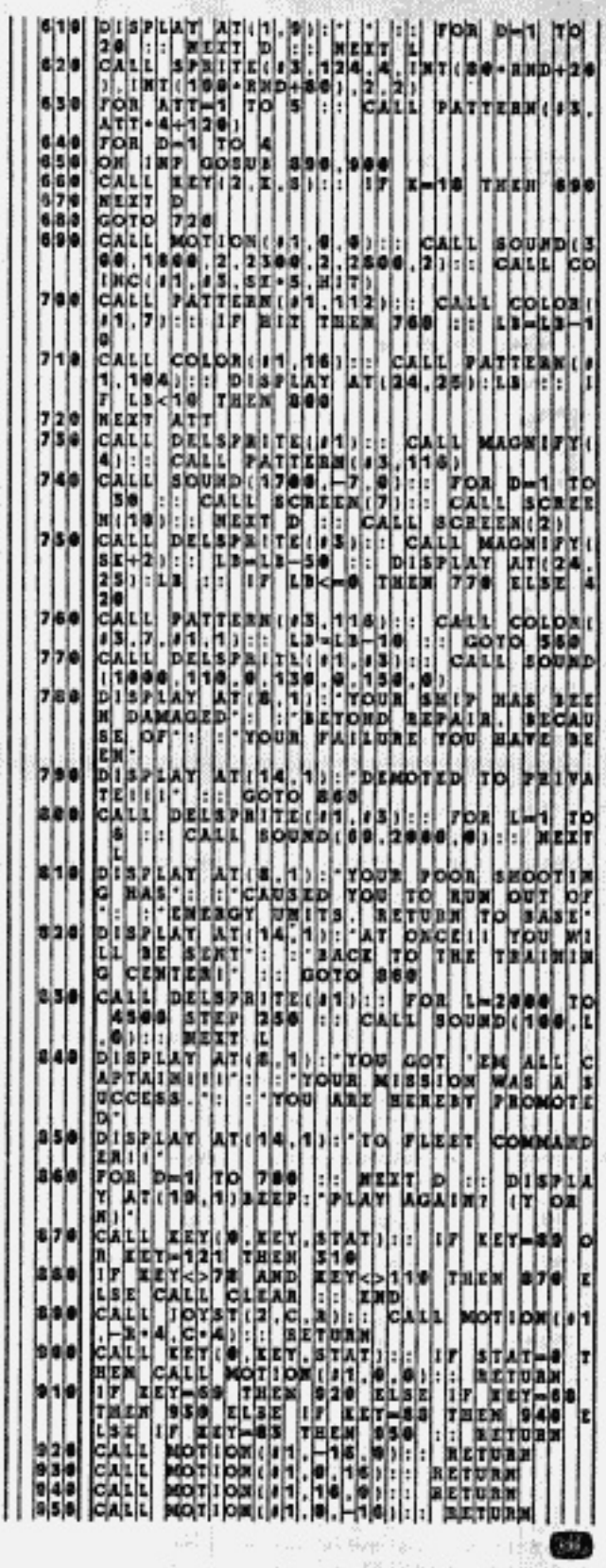
EXPLANATION OF THE PROGRAM

Space Patrol

Line Nos.	
150	Clears screen and makes it black.
160	Sets colors of letters and numbers to white.
170-300	Display title and define characters.
310-330	Clears screen; lets user choose joysticks or keyboard.
340-390	Clears screen; initializes energy and ships destroyed; randomizes; lets user choose high or low skill level; sets magnification 3 for high level, 4 for low level.
400-410	Sets colors for stars; randomly places 40 stars on screen.
420	Creates gun sight in center of screen.
430	Displays energy level and number of ships destroyed at bottom of screen.
440	Randomly selects number from 2 to 6; if the number is 5 or 6, branches to the satellite routine; otherwise a supply ship is defined.
450	Changes colors of stars so they will twinkle.
460-470	Randomly sets speed, direction, and location of supply ship.
480	Branches to joystick or keyboard input to move gun sight.
490-530	Checks if fire button or key is pressed; if so, stops motion of gun sight, makes laser display and sound, checks for a hit, and decreases energy.
540-590	If ship is hit makes red explosion and sound, increments ships destroyed, reduces energy, deletes sprite #3. Checks for end of game and branches.
600-610	Sounds and prints warning for satellite launching.
620-680	Creates satellite sprite and gradually increases the size.
690-720	Moves scope if user indicates or tests for hit if fire button or key is pressed. Energy is reduced by 10 for each shot fired.
730-760	If satellite is hit, it explodes; if not there is a larger blast and energy level is reduced by 50.
770-850	Sounds and messages for end of game depending on number of ships or energy level.
860-880	Displays option to play again.
890	Subroutine to move gun sight with joystick.
900-950	Subroutine to move gun sight with arrow keys.

```
100 REM ****************
110 REM * SPACE PATROL *
120 REM ****************
130 REM
140 REM
150 CALL CLEAR :: CALL SCREEN(2)
160 FOR I=0 TO 8 :: CALL COLOR(I,16,1)
    :: NEXT I
170 DISPLAY AT(9,4)BEEP:"*** SPACE PAT
    ROL ***"
180 CALL CHAR(112,"000000000000000000103010
    0000000000000000000000000000050C0300000
    000000000000000")!SHOT
190 CALL CHAR(116,"0200200421880150001
    28020044080020084400210000024810024
    0040110800")!EXP
200 CALL CHAR(104,"1500400004000040000400
    0400004000015000540000100001000010000
    1000010005400")!GUN SIGHT
210 CALL CHAR(103,"0000000018")!:: CALL C
    HAR(111,"0000000001")!:: CALL CHAR(95,
    "0000000101")!STARS
220 TR$(1)="000000000033488888F8F8848330000
    000000000000000181412F1F2141810000000
    0000"
230 TR$(2)="0141414177F717F7F477F404040
    4000000300282822FE8EFEFEE2FE020202020202
    0000"
240 TR$(3)="00000E0E0E0E0E0E0E3F7FCECE7F
    3F1700000070707070707070FCFE7373FEFC
    F800"
250 TR$(4)="01010101020 7E1915F9F90F16077F
    000000008000004908F91F1F1118F06FE00
    0000"
```





Computer Chess

The game of chess has fascinated men and women for hundreds of years. People from all walks of life and all ages have enjoyed the challenges and entertainment it provides. The universal popularity of chess is undoubtedly due to its resemblance to life: Mastery of chess requires many of the same elements necessary to mastery of one's life—logical thought, long-range planning, the ability to recognize and act on sudden opportunities, persistence, patience, concentration, steady nerves, confidence, objectivity and, of course, lots of experience! Yes, to do well in chess does require all these things, but interestingly enough, practicing the game greatly helps develop and nourish these *same* characteristics and abilities! "Learning to play" is, in reality, one and the same as "playing to learn."

And yet, for all its challenges and self-improvement attributes, the game is enjoyable at all levels of skill—from raw novice to international master. Over the years, chess has provided me with hundreds of hours of engrossing entertainment and many cherished friendships.

With the advent of strong chess-playing computer programs, chess has entered an important new stage of development. People can learn chess much more rapidly than before *without* the often deflating experience of losing many games *in public*. This is especially true for children; losing badly to adults or other children can often drive them from the game. Having a ready and discreet opponent does indeed have its advantages. . .

The TI *Video Chess* Command Cartridge is one of the programs now available. It is a unique implementation since it is contained in 30K of ROM (no time-consuming cassette loading), can run on a "bare-bones" TI-99/4A (no disk drives or other peripherals are needed), uses a keyboard overlay to simplify commands, and has built-in chess clocks. This last feature is useful for users who wish to eventually play in tournaments where the use of chess clocks is mandatory. Playing under tournament conditions is now possible in the privacy on one's own home!

In this initial section, I'll look briefly at the main features [illegible] some of its strengths [illegible] suggestions for using the [illegible] available for using the program. You can (1) play chess against the computer, (2) play against another human opponent, (3) set up a problem for the computer to solve, or (4) have the program play as many as nine (!) opponents simultaneously. In addition, games or positions may be stored on cassette—an especially useful feature for postal players, or players without enough time to finish their games in one sitting.

When playing against the computer, you can control the playing characteristics of the program by choosing the experience level (beginner, novice, or intermediate), the time allotted to the computer for each move (30 seconds to 200 seconds), and the style of play (normal, defensive, or aggressive). The program also allows you to take back a move, ask for advice, have your move evaluated, or even switch sides!

In the problem mode, you can ask the program to solve a checkmate in two, three, or four moves. This is, of course, a potentially valuable learning tool, but the program's versatility doesn't stop there: You can also set up *any* position and have the computer play a normal game starting from the given position.

Based on many years of tournament experience, I would estimate the maximum strength of the program to be slightly less than the average player in a typical chess tournament. This is superior to probably 90 percent of the world's chess players! And presumably, stronger versions of the program will be available in the future. To put this in perspective, the strongest chess-playing program in the world, running on the enormous and fast CYBER or CRAY computers, still does not play at the level of a human chess master. (It will, however, defeat 99 percent of the world's chess players!)

As an educational tool, the *Video Chess* program is excellent. A beginning player can make rapid improvements in his game by adjusting the strength of the program as his own playing strength increases. If you're a new player, you should have at least one good book on chess that is designed for beginners. (There are many good ones on the market.) Then as you learn new ideas and techniques from reading, you can try them out against the computer immediately. For example, it is important for every player to master the basic checkmates: king and queen vs. king; rook and king vs. king; two bishops and king vs. king. All good chess manuals discuss these in detail. After reading about how to mate with king and rook [illegible] mediately try it out using the program [illegible] much faster when a tireless willing [illegible] always ready to play!

The weakest part of the program is in the problem mode when asking for a mate in two, three, or four moves. For example, when I gave the program Problem No. 1A (below), it worked for two and one half hours without coming to a conclusion. I finally turned it off. And I have had similar

disappointing results with rather easy mates in Problem No. 1B. Fortunately, this defect is not terribly important, and may be alleviated in future versions of the program. The best use for this problem-solving mode seems to be in setting up positions from which the computer will commence playing as in a normal game. (Note: You can do this to learn the basic checkmates mentioned previously.)

Finally, I will leave you with two problems to solve. Problem No. 1A comes from a game between two grandmasters about sixty years ago. Problem No. 1B is a famous position—especially memorable because after Black made the beautiful winning move, spectators showered the playing stage with gold coins. As it turned out, however they were *not* showing their admiration. . .but rather, paying off their bets *in disgust!*

Problem No. 1A

White: Pawns: A2, B2, C2, F2, G2, H2
Knights: G1
Bishops: D2, F1
Rooks: H1
Queen: D3
King: C1

Black: Pawns: A7, B7, C6, G7, H7
Knights: B8, E4
Bishop: [illegible]
Rooks: A8, H8
Queen: E5
King: E1

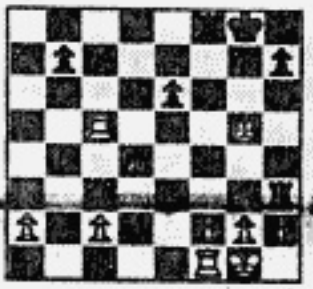
White to move and checkmate in three moves.

Problem No. 1B

White: Pawns: A3, C2, F2, G2, H2
Knights: None
Bishops: None
Rooks: C5, F1
Queen: G5
King: G1

Black: Pawns: A7, B7, E6, G7, H7
Knights: D4
Rooks: B2, F8
Queen: C3
King: G8

Black to move and win
(Black has a single crushing move).



Computer Chess PART TWO

Ever wondered where your computer got the "intelligence" to beat you in a game of chess? It's all in the program, you say? But then where did chess-playing computer programs come from? You might suppose that the impetus for the development of these programs came from chess players themselves. But in fact, this was not the case at all. It was researchers in the field of artificial intelligence (psychologists and computer scientists) whom we have to thank for those embarrassing checkmates. . .

The goal of these reseachers was to determine the nature of intelligence itself: What precisely it was, and consequently, what it was not. This was no easy task. They hoped to shed some light on this problem by getting computers to do things that if performed by a human would require "intelligence." It didn't take long to figure out that chess was a natural: It presumably required highly intelligent behavior, and yet, it was "contained" enough so that initial programs designed just to play "legal" games would not be prohibitively large. As these programs were developed, it soon became obvious that to progress from legal games to good —or even just reasonable—play required close attention to basic theory and concepts as understood by humans. For example, the number of possible positions after only the first ten moves in a game is a number having over a *hundred* zeros in it! Hence, looking at all possible positions is clearly impossible.

As a consequence of this need for a higher level of understanding of the game, strong chess players had to be consulted. One of these was international master David Levy of Scotland. Levy is perhaps best known for his $10,000 bet (made in August 1968) that even within a decade, there still wouldn't be a computer program that could defeat him in a match. In the years since his bet (which he won easily), Levy has been a frequent visitor at computer conferences, where he lectures and plays simultaneous exhibitions against several of the current programs. Incidentally, he also acted as a consultant to Texas Instruments in the development of the *Video Chess* program.

Levy has therefore provided a valuable link between the artificial intelligence commmunity and the large community of chess players. He, perhaps more than anyone else, has been in the position to measure the rate of computer chess progress. In his view (and mine as well), the rather recent advent of microprocessor chess playing machines will make chess popular and accessible as never before. The revolution has just begun!

As indicated above, chess playing programs do not attempt to find a move by searching all posssible combinations of moves. Rather, chess programs combine chess theory and concepts together with brute force searching techniques to choose a move. Therefore, they are limited by how well the program "understands" chess theory and can "think" like a human player, and by speed and memory considerations. The speed and available memory determine how far ahead the program can be examined and evaluated in a given amount of time. The number of moves the program can look ahead in a given position is called its *search horizon* (Levy's term).

For these reasons, even though they play relatively strong chess, chess playing programs have certain characteristic weaknesses which can often be exploited. For example, a program may sacrifice a bishop or a knight on one side of the board to win a rook (with a knight usually) in a corner on the other side. This will leave the knight trapped after it captures the rook. To any human chess player, it would be

evident that the knight was permanently trapped and would eventually be lost—leaving the player with only a rook (5 units) to show for the loss of two minor pieces (a total of 6 units). However, the computer would merely consider the situation a gain of two units (lose a bishop or knight and gain a rook) as long as the stranded knight could not be captured within the number of moves in its search horizon. The limited search horizon leads to other situations where short term expedients are followed to the detriment of position.

Future improvements in speed will extend the search horizon of chess programs and thereby increase their playing strength even further. In my opinion, without considerable improvement in the longer range strategic capabilities of these programs, they will not be able to reach the level of world-class human players. However, we players in the other 99.9 percent had better watch out!

As an experiment, I recently pitted my *Video Chess* (a TI Command Cartridge) program against the Boris machine with the Morphy cartridge. Boris-Morphy is reputedly the strongest comercially available microprocessor chess playing machine. The match consisted of playing the *Video Chess* program at its highest level (Intermediate, 200 seconds per move) against the Boris-Morphy machine at three different levels from high to low. Although the Boris-Morphy program won all three games, the *Video Chess* program did obtain a winning position against the two lower levels (but could not find the knock-out punch). The top level of Boris-Morphy seems clearly stronger than *Video Chess*. All in all, the results were not bad, and since the top current level of *Video Chess* is called "Intermediate," we may look forward to further strengthening of the program.

The two problems I'll leave you with are from games by famous chess players. The first position is from a game of "speed " chess played in 1912 between American Edward Lasker (who died recently at age 96!) and former English champion Sir George Thomas. The rules were, I believe, that neither player could allow his own clock to get more than five minutes ahead of his opponent's clock. To find such a pretty mating combination at that speed is impressive. The second position was played by the great American champion Harry Nelson Pillsbury near the turn of the century in an exhibition where he played blindfolded against 22 different opponents simultaneously! Blindfold play is not as difficult as you might think—try it against your *Video Chess* program sometime—but to play 22 such games successfully is phenomenal. In recent times George Koltanowski has played blindfolded against more than 50 opponents simultaneously. But Pillsury's achievement is magnified by the fact that he could perform well in blind simultaneous play against *masters*!

Problem No. 2A

White: Pawns: A2, B2, C2, D4, F2, G2, H2
Knights: E4, E5
Bishops: D3
Rooks: A1, H1
Queen: H5
King: E1

Black: Pawns: A7, B6, C7, D7, E6, G7, H7
Knights: B8
Bishops: B7, F6
Rooks: A8, F8
Queen: E7
King: G8

White to move and mate in several moves. Can you find the fewest necessary?

Problem No. 2B

White: Pawns: A2, C3, H2
Knights: None
Bishops: E3, E4
Rooks: None
Queen: H4
King: H1

Black: Pawns: A3, B6, C5, H7
Knights: None
Bishops: C6
Rooks: None
Queen: F7
King: H8

Black to move and mate in three moves.

Computer Chess PART THREE

We have discussed the relationship between chess programs and artificial intelligence, and examined some general characteristics of chess playing programs—both strengths and weaknesses. In this article, I'm going to illustrate some of these characteristics through an actual game played between the TI *Video Chess* program and myself. The game was played with the program set on intermediate level, normal mode, with 200 seconds per move allowed.

White: J. Wolfe	Black: TI-99/4 with Video Chess
1. E2 - E4	G7 - G6
2. D2 - D4	F8 - G7

These first two moves constitute the Pirc-Robatsch defense to the opening move E2 - E4. You may have noticed in your play that the program often makes the first several moves quickly and then slows down. This is because certain standard opening sequences are stored

in the program and played automatically in the appropriate situation. As soon as these run out, or as soon as the position is no longer standard, the program reverts to its main programming and hence slows down.

3. B1 - C3	C7 - C6

The main purpose of the opening part of the game is to bring out the pieces and to get a reasonable foothold in the central part of the board. (More precisely, the *center* is the square region whose corners are C3, C6, F6, and F3. The squares D4, D5, E4, and E5 are especially crucial.) Long experience has shown that the success of future maneuvers depends on an adequate control of this area. The last moves for each side fit well into this plan. White brings out a knight that bears down on the center while Black prepares to play D7 - D5 establishing his own foothold there.

4. F1 - C4	B7 - B5
5. C4 - B3	

White develops a piece and temporarily prevents D7 - D5. Black responds by driving back White's bishop and preparing a later pawn advance on the *queen-side* (i.e., the left-hand portion of the board).

5. . . .	D7 - D6

This is a weak move because of the following tactic.

6. C3 - B5

Black cannot capture the knight because White then plays 7. B3 - D5 and captures the rook at A8 coming out with a two unit gain in material. (Recall that a rook is worth 5 units and a bishop 3 units. These units represent the relative strength of the two pieces.)

You might be wondering why the program missed such a short sequence of moves. Well, the reason is fairly complex. The program has two basic features: The first is a *static evaluation feature* which takes a given position and evaluates it to decide which side is better and by how much. This is done by assigning numerical values to certain features of the position and summing these values to get a numerical value for the position. For example, being a pawn ahead in material might be worth, say, 75 points, while not being able to castle (ever) might be worth *minus* 15 points. The program does this for both sides, and the side with the largest score is judged to have the best position. In this evaluation scheme, material advantage is given the largest positive weight by far.

The second basic feature of the program is a *searching procedure*. When combined with the static evaluation program, it allows the program to evaluate the consequences of various moves and to pick what it deduces to be the optimum one. Unfortunately, time and memory considerations limit the number of moves the program can look ahead (i.e., its *search horizon*) and can also limit the number of moves that are considered in response to a contemplated move.

Thus, in examining the position after 5. . . . D7 - D6, White has 38 legal moves. In deciding which moves to consider first as possible replies by White, the program will *not* begin with moves that result in immediate material loss by White. This again is due to the heavy weight assigned to material superiority. Thus the continuation 6. C3 - B5 might not even be reached in the search within the time limit. Sacrifices of material are difficult for all but the most advanced and powerful programs to either make or predict.

6. . . .	D6 - D5

This is a good move and is the other side of the argument above. The program finds the only possible way to regain the lost pawn. Here the emphasis on material is helpful to the program.

7. B5 - C3	D5 - E4
8. C3 - E4	G7 - D4
9. G1 - F3	

Thus, Black has not lost a pawn after all. However, Black's position now has two unpleasant features: First, his pawns at A7 and C6 are weakened since they cannot be protected by pawns if attacked, but must be protected by pieces. This can tie down Black's pieces and will make the pawns vulnerable, especially in the later part of the game when fewer pieces remain. Second, to regain the pawn, Black has exposed his bishop to attack. Thus White can develop a piece (G1 - F3) and at the same time force Black to waste a move either guarding or retreating his bishop. Note that White has three pieces developed and no pawn weaknesses, while Black has only one developed and definite pawn weaknesses. White already has a distinct advantage.

9. . . .	C6 - C5

This is another weak move. Black cannot retreat the bishop to G7 or F6 because of the B3 - F7 check winning the queen, but D4 - B6 is possible—preserving material equality. There is some evidence from this game and others I have played that the search horizon of *Video Chess* is about *two* moves in complicated positions. This would explain why C6 - C5 (so as not to waste a move retreating) was considered best.

10. C2 - C3	D4 - F2 check

Now looking ahead two moves, the program *apparently* can see that if D4 -G7, then B3 - F7 check and White wins the black queen on the next move. However, later when I set up the position after D4 - G7 and asked the program to play White, it played D1 - D8 winning only two pawns (the one on F7 and then the one on C5), leaving White two units ahead. Since giving up a bishop for a pawn also leaves Black two units behind without having to trade queens, the move 10. . . . D4 - F2 was chosen. Thus it appears that the program made the right move for the wrong reason!

11. E1 - F2	G8 - F6

Again the program does not see the third move in the coming sequence.

12. E4 - F6 check	E7 - F6
13. D1 - D8 check	E8 - D8
14. B3 - D5	

Thus White wins another piece.

14. . . .	B8 - C6
15. D5 - C6	A8 - B8

White is so far ahead in material that winning is simple. Accordingly, I will relate the rest of the game with little comment.

16. B2 - B4

This move allows White to play C1 - F4 without an annoying check at the B2 by the black rook.

16.	. . .	C5 - B4
17.	C1 - F4	B8 - B6
18.	A1 - D1 check	D1 - E2
19.	F4 - D6 check	E7 - D8

19. . . . E7 - E6 leads to a quick mate after 20. H1 - E1 check E6 - F5; 21. D1 - D5 check F5 - G4; 22. H2 - H3 mate.

20.	D6 - C5 check	D8 - C7
21.	C5 - B6 check	A7 - B6
22.	C6 - D5	B4 - C3
23.	H1 - E1	C7 - B8
24.	D5 - F7	C8 - G4
25.	E1 - E7	G4 - F3
26.	G2 - F3	F6 - F5
27.	D1 - D7	B8 - C8
28.	F7 - E6	B6 - B5
29.	D7 - A7 check	C8 - D8
30.	E7 - D7 check	D8 - E8
31.	A7 - A8 checkmate	

Currently, the most powerful chess programs can look ahead about *six* moves in fairly complicated positions. Advancements in hardware should extend the capability to *nine* moves. This is about twice as many moves as chess masters can look ahead in complicated positions. Such programs will be virtually impossible to trap in simple tactical sequences and, in fact, the human playtr will most likely be victim. To defeat such a program will require superior application of chess theory and strategy, as well as avoidance of open tactical situations where an eight or nine move look-ahead program would be at its best.

Computer Chess
PART FOUR

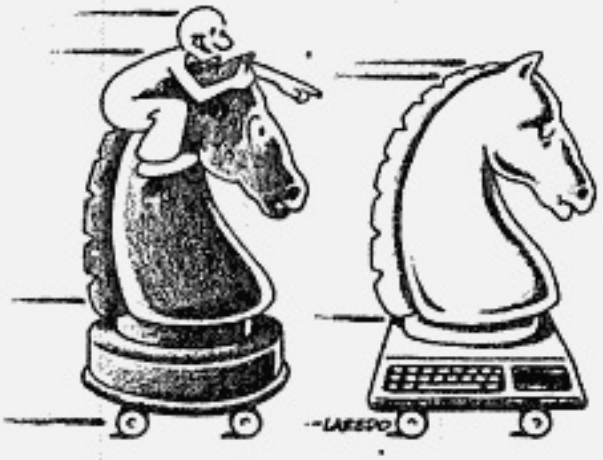
The computer chip has already revolutionized the game and toy industry, and even bigger changes are ahead. Chess playing machines, a specialized branch of this new technology, are now widespread. There are several companies making what are essentially simple microcomputers, completely devoted to playing chess (at least eight companies at last count). This is, of course, in addition to numerous packages of chess software that run on personal computers.

Competition between chess-playing machines has resulted in a continuing strengthening and evolution in performance to the point where there is now a world *micro*computer chess championship held each year. This now complements the [illegible]world computer championship which has been held for [illegible] years, and which features powerful programs, typically requiring large, fast computers.

In September 1981, the second annual World Microcomputer Championship was held in Hamburg, Germany. Four machines competed in the commercial division and eight in the experimental group. The Chess Champion Mark V (Sci Scys, Hong Kong) won the commercial group while the Champion Sensory Challenger (Fidelity, U.S.A.) was a close second and defeated the Mark V in their individual series 2½-1½. In the experimental group, Fidelity Experimental (USA) was first, with Princhess (Sweden) second and a two-way tie for third place between Pilidor Experimental (England) and the Phoenix/Novag Experimental (USA/Hong Kong).

Later in November, the twelfth annual North American Computer Championships were held in Los Angeles. Sixteen programs were entered—from microcomputer programs like Philidor mentioned above, to powerful "move crunchers" like Belle of Bell Labs, and Cray Blitz using the powerful Cray computer that carries out 80 million instructions per second! The winner was the impressive program, Belle (also the world computer champion!) which features, in addition to the basic program, special hardware developed expecially for chess. This program and hardware can examine 23 million (!) chess positions in a three minute period—almost *seven times* as many as its nearest competitor. Finishing in a tie for second were Cray Blitz, Nuchess and Bebe. Even though Cray Blitz is backed up by a computer a hundred times faster than Belle's, it can only examine only about a million positions in a three minute period. This demonstrates the great advantage of having special hardware!

Just in case you're curious about how well these programs play chess, I give you here the crucial last round game between Belle and Cray Blitz for the championship.



Computer Chess
PART FIVE

In this section we are going to look at some variations of standard chess problems, as well as a few interesting challenges associated with chess but not directly related to playing the game. You'll be able to try all of this on your TI-99/4A computer with the *Video Chess* Command Cartridge.

Diversions

By now, of course, you are already well acquainted with chess problems taken from positions in actual games. But chess literature also abounds in problems that have little or no relevance to practical play, but are nevertheless extremely intriguing. Here are a few:

Problem 5A: This is called the "Knight's Tour." Place a knight on an empty board (on A1 for example) and move the knight 63 consecutive times in such a way as to land on each square exactly once and return to the beginning square on the 64th move.

Problem 5B: Remove the squares H1 and A8 from the chessboard. Is the "Knight's Tour" still possible now? You are required to prove that your answer is correct!

Problem 5C: This problem involves a knowledge of chess plus the ability to make logical deductions. While playing a game of chess, Black became irked at his losing position and petulantly removed his king from the board. At that moment, White was in the middle of making his move; for an instant after removal of the black king, the board was completely empty. After White completed his move, Black cooled down and replaced his king. But then he made the worst possible move on the board and White announced mate in two moves. Your task is to reconstruct the position just before White moved and give the exact sequence of moves leading to the checkmate of the black king. (Yes, the problem has a solution.)

Problem 5D: Place eight queens on an otherwise empty board in such a way that no two queens are attacking each other.

Problem 5E: Find the shortest number of moves necessary to produce a *stalemate* starting position.

The above problems represent only a small sample, but perhaps give some idea of the variety of possiblities. Oh yes, I will provide solutions (for all but Problem 5E, for which the minimal number is not known). It is a much smaller number than one would think on first seeing the problem. Try it and see what you can come up with. . .

Versions

Besides the diversions provided by such puzzles, chess players have also been attracted by variations on the basic game of chess. "Speed Chess" (or five-minute chess) is a version requiring a chess clock. Initially each player is given five minutes of time. Play then proceeds until one side is checkmated, a draw is declared, or until one side runs out of time. (For those of you who are not acquainted with the use of a chess clock, I should explain that the player has his clock running until he makes his move. He then pushes a button stopping his own clock and starting that of his opponent.) Thus each game lasts no more than ten minutes. This version is widely popular at chess clubs and among tournament players.

Another currently popular version, especially with younger players, is called "Siamese Chess." This involves four player divided into teams of two players each and requires two chess sets and (usually) two chess clocks. The partners sit on the same side of the table and play opposite colors. Thus the pieces that one partner captures will be the same color as those his partner will be playing on the adjacent board. As one partner captures a piece from his opponent, he passes it to the other partner. The reason for this is that, in addition to the usual moves of chess, one is allowed to place new pieces on the board anywhere that is not occupied—with the one exception that pawns may not be placed on the back ranks (squared A1 - H1 or A8 - H8) where they could be promoted instantly to a more powerful piece.

The placement of new pieces on the board causes the chess battle to take place at an accelerated pace, and causes unusual and often hilarious positions to occur. To make matters worse, the clocks on each board are set for five minutes as in speed chess! The game ends when either a checkmate occurs in one of the games, both games are drawn, or one side runs out of time.

Although chess is far from being "played out," so much study has been devoted to the opening portion of the game that it is possible to go through the first twenty moves in some openings simply repeating moves that are already known to be good. These are called "book moves" because they can be found in chessbooks dealing with openings. This means that a player may obtain a substantial advantage in the opening stages of a game simply by memorizing several sequences of moves found in opening books. At the grandmaster level, this tendency is so refined that victory often hinges on knowing the latest wrinkle in the theory of some particular opening variation, and springing it on a less prepared opponent—one who must then expend extra time on his clock searching for the best reply to this surprise. To combat this over-refinement of opening theory, a simple variation of chess has been proposed. It is called "Prechess" and is played exactly like ordinary chess except for the first eight moves of the game. These proceed as follows: Both sides line up their pawns in the usual way, but leave the row behind the pawns empty. Then the

first eight moves consist of each side alternately (beginning with White, as usual) placing down one piece at a time on the back row anywhere that is unoccupied. This is done until all eight pieces on each side are placed. Then the game continues in the usual way. Since all opening theory is based on the *standard* starting position (which this is usually not), the printed variations found in the opening books are useless.

This version of chess appeals to many serious players and has the advantage that it can be played using a standard set without any bizarre rule changes; basic chess principles still apply as strongly as ever. By using the problem mode of the *Video Chess* program to set up the initial position, you can play "Prechess" on your computers. Try it some time. Some very unusual and interesting games can result from it.

Solutions to Chess Problems

Problem No. 1A:

1. D3 D8 check E8 - D8
2. D2 - G5 double check

(a)2. . . D8 - E8
3. D1 - D8 checkmate
(b)2. . . D8 - C7
3. G5 - D8 checkmate

Problem No. 1B: 1. . . C3 - G3!!

Black appeared to be in trouble since after the apparently forced retreat of his queen out of danger, White could capture the rook on H3 and be decisively ahead in material. Black had foreseen all this, however, and replied with the crushing move above. White has three ways to capture the black queen (which must be captured else mate on H2 is inevitable)—all unsatisfactory.

(a) 2. H2 - G3 D4 - E2 checkmate.
(b) 2. F2 - G3 D4 - E2 check.
3. G1 - H1 F8 - F1 checkmate.
(c) 2. G5 - G3 D4 - E2 check.
3. G1 - H1 E2 - G3 check.
4. H1 - G1 G3 - E2 check.
5. G1 - H1 H3 - C3

and Black is a full piece ahead with an easy win. In the actual game, White resigned after 1. . . C- G3.

Problem No. 2A

1. H5 - H7 check!! G8 - H7
2. E4 - F6 Double check H7 - H6 (the EF - G8 mate)
3. E5 - G4 check
4. F2 - F4 check

(a) 4. . . G5 - F4
5. G2 - G3 check F4 - G5
6. H2 - H4 checkmate or
5. . . F4 - F3
6. 0 - 0 checkmate.

(b) 4. . . G5 - H4
5. G3 - G3 check H4 - H3
6. D3 - F1 check B7 - G2
7. G4 - F2 checkmate

Problem No. 2B

1. . . . F7 - F1 check
2. E3 - G1 F1 - F3 check!
3. E4 - F3 G6 - F3 checkmate.

Problem No. 5A:

Here is a knight's tour beginning at A1. This solution is my own, found after an appropriate amount of trial, error and frustration!

A1 - C2 - E1 - G2 - H4 - F3 - G1 - H3 - F2 - H1 - G3 - F1 -
H2 - G4 - H6 - F7 - H8 - G6 - F8 - H7 - F6 - G8 - E7 - F5 -
G7 - H5 - F4 - E6 - G5 - E4 - D6 - E8 - C7 - A8 - B6 - C8 -
A7 - C6 - D8 - B7 - A5 - C4 - E5 - D7 - B8 - A6 - B4 - A2 -
C3 - D5 - E3 - D1 - B2 - A4 - C5 - D3 - C1 - E2 - D4 - B5 -
A3 - B1 - D2 - B3 - A1 Home Again!

Problem No. 5B:

Removing the square H1 and A8 makes the knight's tour impossible. The reasoning is as follows: Both these squares are the same color (white) so there remain two more black squares than white. But on a knight's tour the color of the squares visited alternates from move to move so that the total number of dark squares and light squares must be the same if a tour is possible.

Problem No. 5C:

It is clear that White must have at least two pieces to checkmate black. The only legal move in which two pieces of the same color may be moved is castling. Then White was in the act of castling when Black removed his own king and hence White has a king and a rook. The remaining problem is to move the black king so that after White castles it can be moved to a square where checkmate in two can be forced (counting the first Black move). A little experimenting leads to the conclusion that the black king is on B3 and White was castling on the queenside. The final sequence is 1. 0 - 0 - 0 B3 - A2 2. D1 - D3 A2 - A1 3. D3 - A3 checkmate!

Problem No. 5D:

It is easy to convince yourself that this one is impossible but it isn't. One solution is to place the queens on A3, B4, C7, D1, E4, F2, G8, and H6.