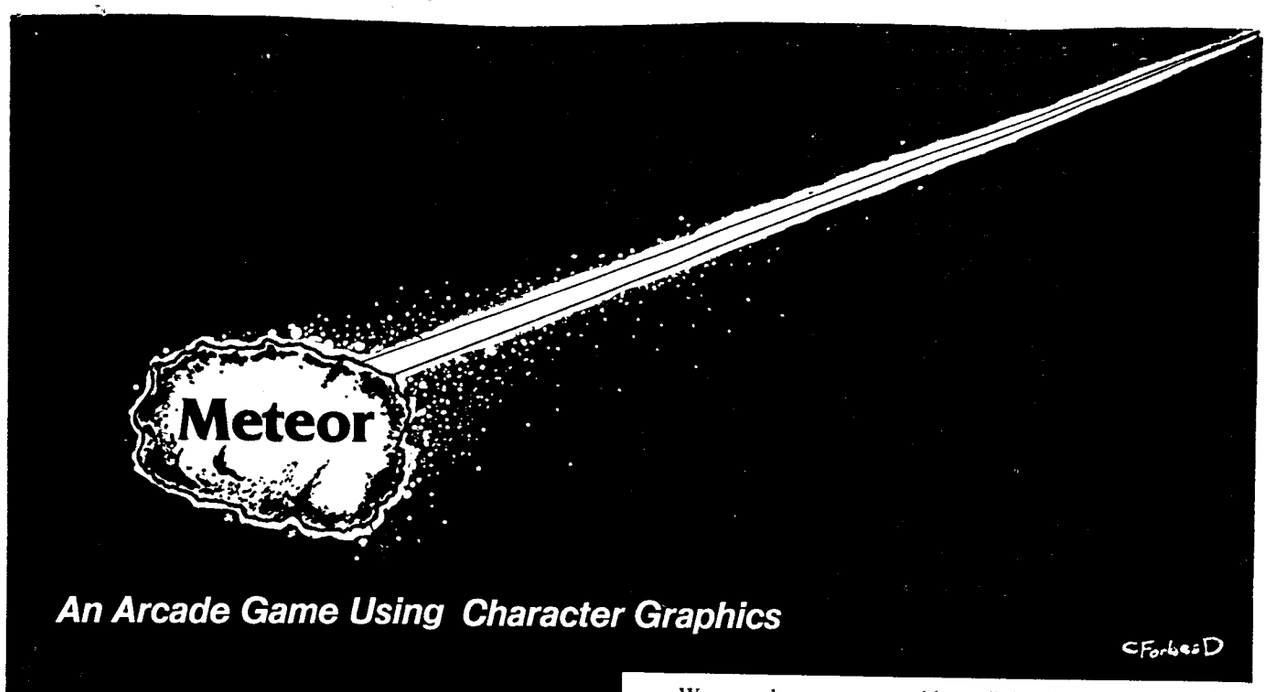



# An Arcade Game Using Character Graphics

Edward T. Ordman

Recently the university for which I work purchased several IBM Personal Computers, and I set about learning how they work. I was familiar with the Apple and TRS-80, so I wanted to check a few new features of the IBM Basic and learn my way around the keyboard and extended character set.

Since our new computers came without the Color/Graphics interface or game paddles/joysticks, a reasonable first project was to determine if an arcade-type game could be constructed using only the keyboard and character graphics on the 25-line by 80-character one-color monitor. The result was surprisingly good, and may be of interest not only to other users of the IBM Personal Computer, but to those with character rather than graphics output in general.

Real life programming jobs are seldom either bottom-up or top-down programming exclusively; one alternates as the program evolves. I started with a bottom-up approach, learning how to do a few things at a time on the new machine.

**Elements of Character Graphics**

First: could I place a mark wherever I wanted on the screen? Yes, and quite easily.

CLS clears the top 24 lines of the screen. It turns out that the 25th line is separate and normally displays descriptions of the ten "programmable function keys." It takes a separate command, KEY OFF, to clear the 25th line.

I decided to display the game in the top 24 lines, and use the 25th for a score display, instructions and the like.

Once the screen is clear, positioning is easy: the command LOCATE Y,X places the cursor in row Y (1 to 25) and column X (1 to 80). Notice that the Y axis is oriented downward in this notation. Now LOCATE Y,X:PRINT C$; will put the character C$ at position X of row Y.

The Basic program crashes if X or Y is out of range, so that is worth testing for, and the computer scrolls if you try to write in the bottom right-hand corner (X=80, Y=24 or Y=25) so a writing routine should protect against that.

Edward T. Ordman, Department of Mathematical Sciences, Memphis State University, Memphis, TN 38152.

We now have a reasonably well-developed point-placing subroutine:

```
330 REM PUT THE CHARACTER C$ AT X(OVER),
    Y(DOWN)
340 IF X<1 OR X>80 OR Y<1 OR Y>24 THEN
    RETURN
350 IF X = 80 AND Y = 24 THEN X = 79
360 LOCATE Y,X : PRINT C$;
370 RETURN
```

Of course, the tests in lines 340-350 can be eliminated if the program tests those conditions before calling the subroutine.

On my way up from the bottom, I asked: can I draw a line? Recalling my analytic geometry, here is a first draft of a way to draw a line from X1,Y1 to X2,Y2, marking each point with character C$:

```
390 REM DRAW A LINE FROM X1,Y1 TO X2,Y2. ASSUME
    Y2>Y1
400 FOR Y = Y1 to Y2
410 X = (X2-X1)/(Y2-Y1) * (Y-Y1) + X1
420 GOSUB 330
430 NEXT Y
440 RETURN
```

A quick experiment reveals that that does not work satisfactorily; X is rarely an integer. Adding 415 X = INT(X + .5) helps, but the loop is still much slower than it should be; we are doing unnecessary repeated calculations during the loop. Here is a better version:

```
390 REM DRAW A LINE FROM X1,Y1 TO X2,Y2 WITH
    Y2>Y1
400 SO = (X2-X1)/(Y2-Y1): S=X1-SO
410 FOR Y=Y1 TO Y2
420 S = S+SO : X = INT(S + .5) : GOSUB 330
430 NEXT Y
440 RETURN
```

This version makes fewer calculations inside the loop.

We could have another version of this program in case Y1=Y2, interchanging X and Y throughout, and maybe even a test for whether ABS(X1-X2) > ABS(Y1-Y2): do we want a line moving nearly horizontally to show as XXX XXX or as X X X ?

Since the latter will give faster motion, I settled for it. On that basis, the routine just given will plot any *descending* line.

These commands and details particular to the IBM Personal Computer are provided so that users of other microcomputers can substitute as may be required by their systems:

*Table 1.*

| | |
|---|---|
| CLS | Clear Screen. This applies only to lines 1 - 24 unless KEY OFF is in effect. |
| KEY OFF, KEY ON | Turns off and on the line 25 display of meanings of the 10 programmable keys. |
| LOCATE A,B | Moves the cursor to line A (range 1-25), position B (range 1-80). |
| SCREEN(A,B) | Returns an integer: the number of the character presently appearing at line A, position B, on the screen. |
| STRING$(N,K) | Returns a character string N characters long, each character is character number K. |
| SOUND F,B | Sounds the speaker, frequency F, for duration B units. |
| INKEY$ | The one or two character long character string denoting the most recently pushed key, or the empty string if no key has been pushed since it was last referenced. |
| RANDOMIZE N | Restart random number sequence, based on the seed N. An unpredictable N may be obtained by extracting substrings from TIME$. |
| TIME$ | The clock time since the system was booted up, as a character string, of the form 02-25-14 for 2 hours, 25 minutes, 14 seconds. May be reset to actual time, if desired. |

Screen Characters used:

| | |
|---|---|
| 2 | Bright Face (1 is Dark Face) |
| 25 | Down arrow, ↓ |
| 219 | Full box (all white if writing white on black) |
| 178 | Shaded box (grey, if writing white on black) |

Keyboard Characters used:

| | |
|---|---|
| 13 | Enter, often called carriage return. Denoted ↵ on key. |
| 32 | Space or blank |
| 0-77 | Cursor right returns a two-character string, CHR$(0)+CHR$(77) |
| 0-75 | Cursor left |
| 0-72 | Cursor up |
| 0-80 | Cursor down |
| 0-83 | DEL (Delete) |
| 0-82 | INS (Insert) |

**A Top-Down Approach**

With those subroutines in hand. I soon had a slow shower of characters falling down my CRT screen. A game began to take shape in my mind.

## A Note On RANDOMIZE

IBM Personal Computer Basic will produce the same sequence of random numbers each time unless you use the command RANDOMIZE. You must provide a "seed," or starting value, in the range -32767 to 32767. Of course, you would like that seed to be unpredictable, and if possible different almost every time you start. Here are two methods:

- In this game there is a keyboard input called for very early: DO YOU WANT DIRECTIONS? In the loop at lines 160-170, we repeatedly test (using INKEY$) to see if the Y, N, or Enter key has been struck. The variable R is incremented each time we go around that loop, and kept in the range 0 to 32003 by the MOD (modular arithmetic) addition. Thus the seed in RANDOMIZE R depends on how quickly the user pressed the key.
- A somewhat easier method is available in Disk Basic or Advanced Basic, since these have the keyword TIME$. One and a half minutes after the computer has been turned on, if the user has not reset it, this variable has the value "00-01-30" (No hours, one minute, thirty seconds). We can convert it to a number and use it as a seed by a command such as RANDOMIZE VAL(MID$(TIME$,7,2)+MID$(TIME$,4,2)).This will produce RANDOMIZE 3001 at the just-mentioned time after startup. While this method is certainly easier than the first method, I have included the first method in the listing since the second method is unavailable in cassette Basic.

**Trial Program Outline (Version 1)**

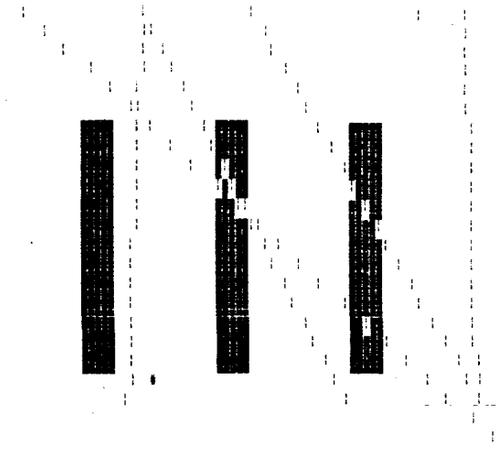
1. Clear screen.
2. Create a marker showing where the "player" is.
3. Periodically have objects falling from the top of the screen.
4. Give the player a way to move his marker, to dodge the falling objects.
5. Turn ends when the player's marker is hit by a falling object.

This was not entirely satisfactory. It would be far more satisfactory if there were something the player could do (other than simply survive) to score points.

Basic character graphics run a bit slowly to allow shooting sorts of games, which I don't really like very much anyway. I decided instead to try letting the player marker move through a path or obstacle course, or gather points when it got to certain targets.

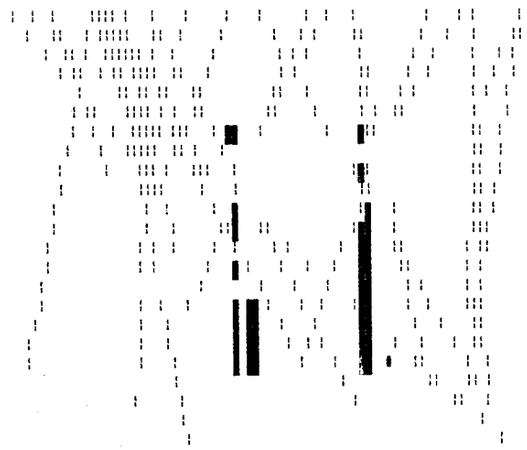
**Program Outline (Version 2)**

1. Clear screen; position player marker.
2. Create targets on screen.
3. Choose a path for a falling "meteor."
4. Each time the meteor falls one position:
   5. If it hits the player marker, go to step 13 (end of turn).
   6. Erase old meteor position, mark new one.
   7. Does player want to move his marker? (Read keyboard) If he does:
      8. If new position is occupied, perform step 12 (score).
      9. Erase old position, mark new one.
   10. If meteor has further to fall, go to step 4.
11. Return to step 3.
12. (Score) Depending on the target hit, increment score, display score on the screen, make a noise: return to main program.
13. (Player is hit). Sound a noise. Draw explosion. Await input to decide whether to play again (go to step 1) or exit program.



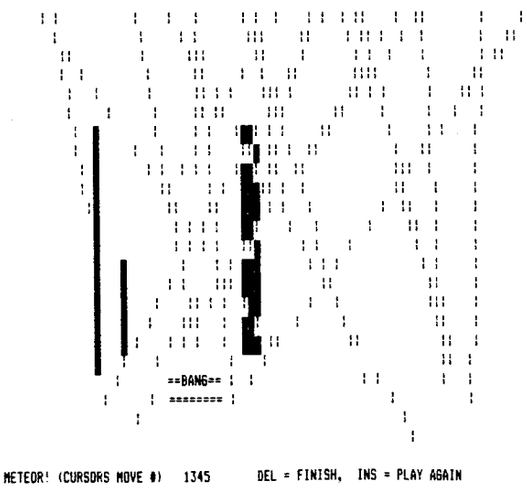
```
METEOR! (CURSORS MOVE #)  -9      KEYS: INS=CONTINUE, DEL=STOP, ENTER=RESTORE
```

*Early in the game.*



```
METEOR! (CURSORS MOVE #)  1346   KEYS: INS=CONTINUE, DEL=STOP, ENTER=RESTORE
```

*Well into the game—many of the targets have been erased.*




```
METEOR! (CURSORS MOVE #)  1345      DEL = FINISH,  INS = PLAY AGAIN
```

*End—A Meteor has hit the player.*

*Here are some pictures showing the game Meteor being played. The pictures were made using an Epson MX-80 printer, which requires a somewhat different character set than is used on the CRT screen. The meteor trails are shown here as short vertical marks (¦) and the player's position is marked with a #.*

```
                                METEOR

A SIMPLE ARCADE GAME USING CHARACTER GRAPHICS.

THE CURSOR CONTROL KEYS START THE # SYMBOL MOVING.
THE SPACE BAR STOPS ALL ACTION TEMPORARILY, AND ALLOWS RESTORING TARGETS.
ANY LETTER (AND SOME OTHER KEYS) WILL STOP CURSOR MOTION.

SEE IF YOU CAN ERASE THE SOLID BLOCKS BEFORE A FALLING METEOR HITS YOU.
EACH ■ YOU ERASE SCORES 10 POINTS, EACH ¦ 2 POINTS.
YOU LOSE 1 POINT FOR EACH ■ A METEOR HITS.

TO HIT YOU A METEOR NEEDS TO GET WITHIN THE SHADED AREA:

                                  ======
                                  ==#===
                                  ======

SOME EXTRA INSTRUCTIONS WILL BE ON THE BOTTOM LINE

HOW HARD (1-9)?
```

*The directions as shown on the screen (symbols altered for printer).*

## Converting Graphics for Printer Output

I have an unmodified Epson MX-80 printer. Its graphics characters are in many ways better suited to the TRS-80 than to the IBM Personal Computer. Still, I wanted to print enough to show what the screen looked like while playing the game. Accordingly, to make the screen printouts provided with this article, I did the following:

For the characters used in the program, I substituted characters that would have a similar general visual effect on the Epson printer. This involved changing lines 120, 140, 370, 700, and 710. The substitutions I made were as follows:

For CHR$(2) (face) I used CHR$(35), that is, #.

For CHR$(219) (solid square) I used CHR$(223), the printer solid block.

For CHR$(25) (down arrow) I used CHR$(124), the vertical line ¦.

For CHR$(178) (shaded square) I used CHR$(61), the equal sign =.

Then I set the printer in *compressed character* (132 characters per line) mode to make the printed shape close to the original screen shape. I ran the game, periodically using the pause-on-space-bar feature of the game to halt the action, and print out the screen using the built-in Print Screen key.

### Built-In Machine Functions

It is now clear that we need to know at least two more functions: how to find out what is displayed at a point on the screen, and how to detect keyboard input without waiting for an INPUT statement.

The first function is easy: SCREEN(Y,X) is a built-in numeric function which returns the number of the character present on the screen in position X of line Y. (In some versions of Basic, on other computers, this may take a PEEK or other technique. In extreme cases—a computer talking to a very dumb terminal—it may require keeping a copy of the screen in an array in memory.)

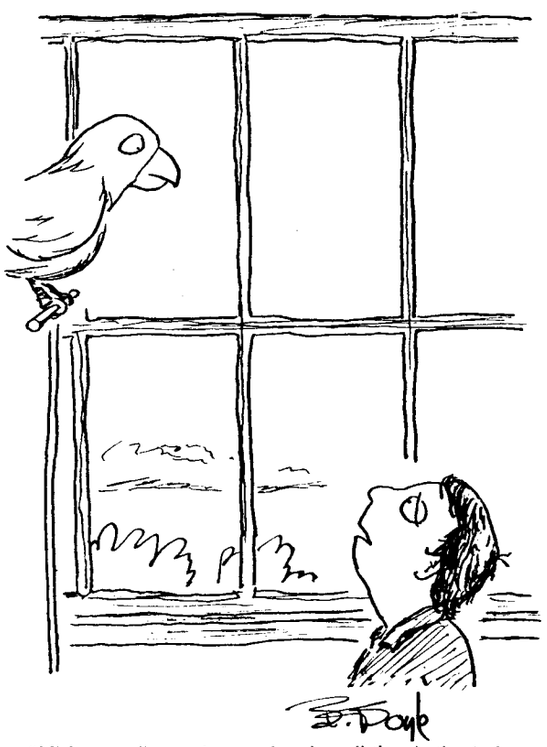
The second is also easy: INKEY$ is a reserved word whose value is the key recently pressed. There is a slight complication, however: on the IBM Personal Computer, some keys produce a two-character value for INKEY$. For example, the Home button produces CHR$(0)+CHR$(71) for INKEY$.

From here on, it is largely a matter of picking specific characters for attractive graphics, fine tuning, and "dressing up" the game. The IBM Personal Computer provides a nice range of graphics symbols, including two small faces, one of which I selected for the player marker, and a downward arrow which was ideal for a falling meteor.

To create more scoring activity, I deducted a point when a meteor hit a target, and left meteor trails on the screen, awarding points when the player erases them. I put in a pause feature controlled by the space bar and a provision for restoring the targets when they were all erased.

### Survey of the Code

A listing of the program is provided in Listing 1. Since the code, including directions and remarks, is under 100 lines, it should be easy to alter. The relatively modular design makes it easy to change into games having little superficial resemblance to the one shown here. For example, changing lines 840-900 would rearrange the targets; an alteration in lines 280-310 or 390-430 could change the character of motion of the attackers or meteors. It would be quite easy to install obstacles, that is, points through which the player marker, meteors, or both could not pass; testing that the player marker is not hitting such an obstacle would go in at around lines 700-720.



*"Might as well cancel your ad; today, all the pirating is done electronically."*

A description of the code, with references to the Program Outline (Version 2) above, follows:

Line 120 clears the screen and initializes.

Line 150 offers directions. If wanted, GOSUB 930.

Line 180 randomizes. IBM Basic requires a seed; see box for discussion.

Line 190 asks how hard it should be (move targets up or down screen). Note: Hitting Enter alone causes defaults: no directions, difficulty = 5.

Line 230 chooses starting location HX,HY for the player marker.

Line 240 clears the screen (step 1 of outline).

Line 250 places instructions in the bottom line.

Line 260 uses GOSUB 840 to place targets on the screen (step 2).

Lines 270-320 choose a path for the falling meteor (step 3, 11); GOSUB 390 to actually plot the line.

Line 330 (subroutine) plots a point for the meteor; checks for player marker hit and for keyboard input. If H$ is not null, but K$ is, continue player marker motion as before (step 5-6-7).

Line 390 (subroutine) draws a line for the falling meteor. If the meteor is below the player marker and cannot hit it, terminate line (step 4).

In line 450, the player is hit by a meteor. The program waits for the player to hit the INS(ert) key to restart, or the DEL(ete) key to exit (step 13).

Lines 570-730 process requests for player marker movement (step 8-9).

Line 740 increments score (step 12).

Line 760 processes pauses resulting from depression of space bar. It offers choices of continuing, exiting program, restoring targets.

Line 840 prints targets and places player marker on screen.

Line 930 gives directions. □

## Two Minor Nuisances

I encountered two minor nuisances that I would regard as slight criticisms of the hardware and software design for the IBM Personal Computer.

- The cursor control keys double as the numeric keypad; the Numeric Lock switch alters their function. If you accidentally strike this key during repeated use of this keypad, strange things happen. In the case of this game, when you are striking cursor control keys repeatedly, hitting Numeric Lock freezes the player marker on the board and inhibits response to the cursor control keys. (This could be overcome in the game design).
- When debugging, I frequently LIST a program and stop the scrolling to read a portion of the listing; scrolling is stopped on the IBM Personal Computer by Control-Numeric Lock. Unfortunately, this can freeze the screen *in the middle* of a single scroll, causing one line of the screen (the one being rewritten) to contain false information about the program contents. (The screen will regenerate, correcting the error, if you try to exit the listing to fix the error.)

*Listing 1.*

```
100 REM  METEOR, A CHARACTER GRAPHICS ARCADE GAME
110 REM  BY EDWARD T. ORDMAN      NOVEMBER 1981
120 H$=CHR$(2):C$=CHR$(219):X$=CHR$(25):REM FACE,SOLID SQUARE,DOWN ARROW
130 C5$=C$+C$+C$+C$+C$:H$="":T=0:REM BLOCK,LATCH FOR FACE MOTION,SCORE
140 Y=178:E2$=STRING$(2,Y):E5$=STRING$(5,Y):E8$=STRING$(8,Y):REM SHADING
150 CLS:KEY OFF:PRINT "DO YOU WANT DIRECTIONS (Y/N)?":R=523:REM RANDOM SEED
160 R$=INKEY$:IF R$="Y" THEN GOSUB 930:GOTO 180
170 IF R$="N" OR R$=CHR$(13) THEN 180 ELSE R=(R+511)MOD 32003:GOTO 160
180 RANDOMIZE R:REM SEED BASED ON DELAY IN ANSWERING QUESTION
190 PRINT "HOW HARD (1-9)?";
200 R$=INKEY$:C=ASC(R$+" "):IF C>48 AND C<58 THEN C=C-48:GOTO 230
210 IF C=13 THEN C=5 ELSE 200
230 HX=20+INT(40*RND+1):HY=16+INT(9*RND+1):REM POSITION FOR FACE
240 CLS:LOCATE 25,1:PRINT "METEOR! (CURSORS MOVE ";M$;")";
260 GOSUB 840 :REM PUT TARGETS,PLAYER MARKER
280 Y1=1:Y2=24  :REM DESCRIBE METEOR PATH
290 X1=INT(RND*80+1):X2=INT(RND*80+1):REM EACH IS INTEGER 1-80
310 GOSUB 390:GOTO 290:REM PLOT METEOR PATH, REPEAT
330 REM PLOT X$ AT Y,X CHECKING FOR SCORES, FACE MOTION
340 K$=INKEY$:IF K$<>"" THEN H$=K$:REM H$ IS LATCH
350 IF LEN(H$)>0 THEN GOSUB 570 :REM KEY WAS STRUCK
360 IF ABS(X-HX)<3 AND ABS(Y-HY)<2 THEN 450 :REM FACE HIT
370 IF SCREEN(Y,X)=219 THEN C2=-1:SOUND 660,2:GOSUB 740:REM TARGET HIT
375 IF Y=24 AND X=80 THEN X=79 :REM WRITING 24,80 CAUSES SCROLLING
380 LOCATE Y,X :PRINT X$;:RETURN
390 REM DRAW A LINE FROM  X1,Y1 TO X2,Y2
400 S0=(X2-X1)/(Y2-Y1):S=X1-S0
410 FOR Y=Y1 TO Y2: S=S+S0: X=INT(.5+S)
420 IF Y>HY+1 THEN RETURN:REM GIVE UP IF BELOW TARGET
430 GOSUB 330:NEXT Y :RETURN
450 REM  TARGET IS HIT, POSITION MESSAGE
460 HX=HX-4:IF HX>72 THEN HX=72
470 IF HX<1 THEN HX=1
480 IF HY=24 THEN HY=23
500 SOUND 400,8:LOCATE HY,HX:PRINT E2$+"BANG"+E2$;:LOCATE HY+1,HX:PRINT E8$;
520 LOCATE 25,35:PRINT "    DEL = FINISH.  INS = PLAY AGAIN          ";
530 H$=INKEY$:IF H$=CHR$(0)+CHR$(83) THEN CLS:KEY ON:END
540 IF H$=CHR$(0)+CHR$(83) THEN CLS:KEY ON:END
550 IF H$=CHR$(0)+CHR$(82) THEN CLS: RUN
560 GOTO 530
570 REM PROCESS KEYBOARD REQUEST
580 IF H$=CHR$(32) THEN 760:REM PAUSE ON SPACE BAR
590 IF LEN(H$)=1 THEN H$="":RETURN
600 HH=ASC(RIGHT$(H$,1)):K$=H$:H$="":LOCATE HY,HX:PRINT " ";
630 IF HH=77 THEN HX=HX+1:H$=K$:IF HX>80 THEN HX=1
650 IF HH=75 THEN HX=HX-1:H$=K$:IF HX<1 THEN HX=80
670 IF HH=80 AND HY<24 THEN HY=HY+1:H$=K$
680 IF HH=72 AND HY>1 THEN HY=HY-1:H$=K$
690 IF HX=80 AND HY=24 THEN HY=23
700 IF SCREEN(HY,HX)=219 THEN SOUND 440,1:C2=10:GOSUB 740
710 IF SCREEN(HY,HX)=25 THEN SOUND 420,1:C2=2:GOSUB 740
720 LOCATE HY,HX:PRINT H$;:RETURN
740 T=T+C2:LOCATE 25,27:PRINT T;:RETURN:REM SCORE POINTS
760 LOCATE 25,35:PRINT "KEYS: INS=CONTINUE, DEL=STOP, ENTER=RESTORE  ";
770 H$=INKEY$:IF H$=CHR$(0)+CHR$(82) THEN 910
790 IF H$=CHR$(13) THEN 840
800 IF H$=CHR$(0)+CHR$(83) THEN CLS:KEY ON:END
810 GOTO 770
840 REM  PUT TARGETS AND FACE ON SCREEN
860 FOR I=12-C TO 24-C
870 LOCATE I,15:PRINT C5$;:LOCATE I,35:PRINT C5$;:LOCATE I,55:PRINT C5$;
900 NEXT I:LOCATE HY,HX:PRINT H$;
910 LOCATE 25,35:PRINT "     HIT SPACE BAR TO PAUSE                ";
920 RETURN
930 REM  DIRECTIONS
940 CLS:PRINT:PRINT TAB(35);"METEOR":PRINT:PRINT
950 PRINT "A SIMPLE ARCADE GAME USING CHARACTER GRAPHICS."
960 PRINT :PRINT "THE CURSOR CONTROL KEYS START THE ";H$;" SYMBOL MOVING."
970 PRINT "THE SPACE BAR STOPS ALL ACTION TEMPORARILY, AND ALLOWS ";
975 PRINT "RESTORING TARGETS."
980 PRINT "ANY LETTER (AND SOME OTHER KEYS) WILL STOP CURSOR MOTION."
990 PRINT:PRINT "SEE IF YOU CAN ERASE THE SOLID BLOCKS BEFORE A FALLING ";
995 PRINT "METEOR HITS YOU."
1000 PRINT "EACH ";C$;" YOU ERASE SCORES 10 POINTS, EACH ";X$;" 2 POINTS."
1010 PRINT "YOU LOSE 1 POINT FOR EACH ";C$;" A METEOR HITS."
1020 PRINT:PRINT "TO HIT YOU A METEOR NEEDS TO GET WITHIN THE SHADED AREA:"
1030 PRINT:PRINT TAB(37);E5$:PRINT TAB(37):E2$+H$+E2$
1050 PRINT TAB(37);E5$:PRINT:PRINT
1060 PRINT "SOME EXTRA INSTRUCTIONS WILL BE ON THE BOTTOM LINE":PRINT:RETURN
```