PROGRAMMER'S NOTEBOOK

By Jim Keogh

Fire That Gun!

PROFESSIONAL systems designers are frequently able to design software packages quickly by using the modular approach to programming. They develop software packages from existing program subroutines. A subroutine is a set of computer instructions which can be used throughout the program by "calling" it. In BASIC, a subroutine is "called" in the program simply by inserting GOSUB, with the line number of the subroutine, wherever the subroutine is needed. At the end of the subroutine, the statement RETURN is used to automatically return the program to the line immediately following the GOSUB statement. Subroutines need only be *keyed* into the program once, usually at the end of the program. (Lines 10,000 and above are good places for storing subroutines.) They can also be *inserted* into the program any number of times using the GOSUB statement each time.

When a programmer designs a new software package, existing subroutines are used with slight modifications to develop the final program.

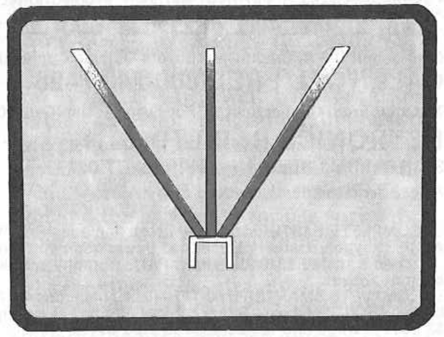
In this column, we will present material that can serve as the basis for your own subroutine file. From the file you can modify subroutines and build your own program. In each column, we will present subroutines and tell you how they operate. By picking apart a subroutine, you will develop a better understanding as to how to modify it to meet your needs.

Subroutines which we will talk about will be written in BASIC. Since it is nearly impossible to write a single program which will run on every microcomputer, we will concentrate on the TRS-80 and the Apple II microcomputers. There are similarities in the BASIC used by other memory-mapped video microcomputers and with little effort you should be able to modify the subroutines presented here to run on your microcomputer. The graphics subroutines, however, cannot be modified to run on serial-port microcomputers unless an "intelligent" terminal and its particular commands are used.

Our first subroutine is one that allows you to fire a shot within the range of a "gun" on your computer screen. Table I is an example of a "gun firing" subroutine which can be used when developing your own microcomputer game.

To use this subroutine, key it into your microcomputer. When you call the subroutine, you will be requested to aim the gun once the gun has been positioned on the screen. INPUT of any number between −1 and +1 will fire the gun. Zero fires it straight ahead. Using −1 will fire a shot to the far left, and using 1 (you need not use the plus sign before the number) will fire a shot to the far right. Use numbers less than 1 and more than −1 (decimals) to fire shots between the gun's horizontal limits.

Let's take a closer look at the subroutine. Starting with line 30, the gun is drawn on the screen. It's not much of a gun, but you can use your artistic talents to improve the gun's looks. You can position the gun at any location on the screen by changing the coordinates of the graphic statement. However, for better results it is important that the horizontal portion of the gun have a center block. Always use an odd number of blocks when drawing this portion of the gun. In the example, five horizontal blocks were lit, making the third block the center. This will become important when we fire the gun.


The gun, when drawn on your screen and . firing, should look something like this.

Line 330 is the input statement. It is important that the input remain within the horizontal range of the gun; otherwise an error message may occur.

Now it is time to fire the gun. Note line 370. This coordinate is the center block of the gun. It is from this location that the "bullet" will "leave" the gun. You can see why it is important to have a center block on the horizontal portion of the gun. Otherwise you may be firing the shot from the corner of the gun instead of the center.

An important part of the subroutine is the beginning statement of the loop that displays the shot. The loop starts with line 390. This instruction states that, for the TRS-80, D = 33 to 2 STEP −1, and for Apple II, D = 33 to 10 STEP −1.

The 33 is the horizontal coordinate of the gun. The 2 (10 for Apple II) is the vertical range of the gun. This is the last vertical block which the bullet will "hit." The STEP −1 statement determines the speed of the bullet.

You can easily modify this subroutine and make the 2 (10 for the Apple II) a variable which the player can control (the range) and the speed of the bullet, by changing STEP −1. Here, −1 is slow and −6 is extremely fast. Numbers between −1 and −6 will give the player a corresponding increase in speed.

When you use the subroutine in your program, be sure to remove our many notes contained under the statements REM. They were only used to provide a better understanding of the functions of each statement. Also, line numbers should be changed to conform to your own programs. We used line numbers in our subroutine and even skipped a few line numbers in our Apple II subroutine to make it easier for you to follow our discussion.

If you have a color display terminal, be sure you modify the subroutine to brighten up the action. When you change from a black-and-white screen to a color display, you also may see some graphic changes in our subroutine, depending upon the kind of microcomputer you use.

With some imagination you can modify this subroutine and add targets of all types which the player must shoot. You can also reverse the coordinates and position the gun at the top of the screen—in an "aircraft"—and shoot at ground targets. ◇

TRS-80 SUBROUTINE

```
 10 REM CLEAR SCREEN
 20 CLS
 30 REM DRAWS LEFT SIDE OF GUN. A= NUMBER OF VERTICAL
 40 REM SCREEN POSITIONS WHICH SHOULD BE TURNED ON.
 50 REM B = THE NUMBER OF HORIZONTAL POSITIONS FROM
 60 REM THE LEFT SIDE OF THE SCREEN FROM WHICH THE
 70 REM VERTICAL SCREEN POSITIONS SHOULD START.
 80 FOR A = 33 TO 35
 90 B = 53
100 SET (B, A)
110 NEXT A
120 REM DRAWS TOP OF GUN. A = NUMBER OF HORIZONTAL
130 REM SCREEN POSITIONS WHICH SHOULD BE TURNED ON.
140 REM B = THE NUMBER OF VERTICAL POSITIONS FROM
150 REM THE TOP OF THE SCREEN FROM WHICH THE
160 REM HORIZONTAL SCREEN POSITIONS SHOULD START.
170 FOR A = 53 TO 57
180 B = 33
190 SET (A, B)
200 NEXT A
210 REM DRAWS RIGHT SIDE OF THE GUN. A = THE NUMBER
220 REM OF VERTICAL SCREEN POSITIONS WHICH SHOULD BE
230 REM TURNED ON. B = THE NUMBER OF HORIZONTAL
240 REM POSITIONS FROM THE LEFT SIDE OF THE SCREEN
250 REM FROM WHICH THE VERTICAL SCREEN POSITIONS
260 REM SHOULD START.
270 FOR A = 33 TO 35
280 B = 57
290 SET (B,A)
300 NEXT A
310 REM AIM GUN. INPUT STATEMENT FOR DIRECTION OF GUN.
320 PRINT @ 64, "AIM GUN SELECTION +/− 1"
330 INPUT A
340 REM THIS STATEMENT REMOVES THE INPUT STATEMENT FROM
350 REM THE SCREEN.
360 PRINT @ 64, "                              "
370 REM FIRE GUN. C = THE CENTER POSITION OF THE GUN.
380 C = 55
390 REM SPEED OF SHOT. D = THE VERTICAL SCREEN POSITION
400 REM OF THE SHOT. 33 = THE VERTICAL STARTING POSITION
410 REM OF THE SHOT. 2 = THE END VERTICAL POSITION OF THE
420 REM SHOT. −1 = THE NUMBER OF VERTICAL POSITIONS FROM
430 REM 33 TO 2 WHICH SHOULD BE TURNED ON.
440 FOR D = 33 TO 2 STEP −1
450 REM C = C + A MEANS THE INPUT QUANTITY (+/−1) IS ADDED
460 REM TO THE VALUE OF C WHICH IS 55, THE CENTER
470 REM HORIZONTAL SCREEN POSITION OF THE GUN. THE RESULT
480 REM WILL GIVE THE NEW STARTING HORIZONTAL SCREEN
490 REM POSITION FROM WHERE THE SHOT IS FIRED.
500 C = C+A
510 REM THE SCREEN POSITION IS TURNED ON.
520 SET (C,D)
530 REM HERE IS THE TIMING LOOP WHICH DETERMINES HOW
540 REM LONG THE SCREEN POSITION IS TURNED ON.
550 FOR T = 1 TO 15
560 NEXT T
570 REM HERE IS WHERE THE SCREEN POSITION IS TURNED OFF.
580 RESET (C,D)
590 REM THIS TELLS THE COMPUTER TO GO TO THE NEXT VERTICAL
600 REM SCREEN POSITION AND TURN IT ON. THE END OF THE
610 REM LOOP.
620 NEXT D
630 REM IF YOU INCLUDE THIS SUBROUTINE AS PART OF YOUR
640 REM PROGRAM, CHANGE LINE 630 TO READ 630 RETURN.
```

APPLE II SUBROUTINE

```
 10 REM CLEAR SCREEN
 20 CALL −936
 30 REM DRAWS LEFT SIDE OF GUN. THE VERTICAL LINE
 40 REM IS DRAWN FROM VERTICAL SCREEN POSITION 33
 50 REM TO VERTICAL SCREEN POSITION 35. 18 INDICATES
 60 REM THE HORIZONTAL SCREEN POSITION FROM WHICH
 70 REM THE VERTICAL SCREEN POSITION SHOULD START.
 80 VLIN 33, 35 AT 18
120 REM DRAWS THE TOP OF GUN. THE HORIZONTAL LINE
130 REM IS DRAWN FOR THE HORIZONTAL SCREEN POSITION
140 REM 18 TO HORIZONTAL SCREEN POSITION 22.33
150 REM INDICATES THE VERTICAL SCREEN POSITION FROM
160 REM WHICH HORIZONTAL SCREEN POSITION SHOULD START
170 HLIN 18,22 AT 33
210 REM DRAWS THE RIGHT SIDE OF THE GUN. THE VERTICAL
220 REM LINE IS DRAWN FROM VERTICAL SCREEN POSITION 33
230 REM TO VERTICAL SCREEN POSITION 35. 22 INDICATES
240 REM THE HORIZONTAL SCREEN POSITION FROM WHICH
250 REM THE VERTICAL SCREEN POSITION SHOULD START.
270 VLIN 33,35 AT 22
310 REM AIM GUN. INPUT STATEMENT FOR DIRECTION OF GUN.
320 PRINT "AIM GUN SELECT +/−1"
330 INPUT A
370 REM FIRE GUN. C = THE CENTER POSITION OF THE GUN.
380 C = 20
390 REM SPEED OF SHOT. D = THE VERTICAL SCREEN POSITION
400 REM OF THE SHOT. 33 = THE VERTICAL STARTING POSITION
410 REM OF THE SHOT. 10 = THE END VERTICAL POSITION OF
420 REM THE SHOT. −1 = THE NUMBER OF VERTICAL POSITIONS
430 REM FROM 33 TO 10 WHICH SHOULD BE TURNED ON.
440 FOR D = 33 TO 10 STEP −1
450 REM C = C+A MEANS THE INPUT QUANTITY (+/−1) IS ADDED
460 REM TO THE VALUE OF C WHICH IS 20, THE CENTER
470 REM HORIZONTAL SCREEN POSITION OF THE GUN. THE RESULT
480 REM WILL GIVE THE NEW STARTING HORIZONTAL SCREEN
490 REM POSITION FROM WHERE THE SHOT IS FIRED.
500 C = C+A
510 REM THE SCREEN POSITION IS TURNED ON.
520 PLOT C,D
590 REM THIS TELLS THE COMPUTER TO GO TO THE NEXT VERTICAL
600 REM SCREEN POSITION AND TURN IT ON. THE END OF THE
610 REM LOOP.
620 NEXT D
630 REM IF YOU INCLUDE THIS SUBROUTINE AS PART OF YOUR
640 REM PROGRAM, CHANGE LINE 630 TO READ 630 RETURN.
```