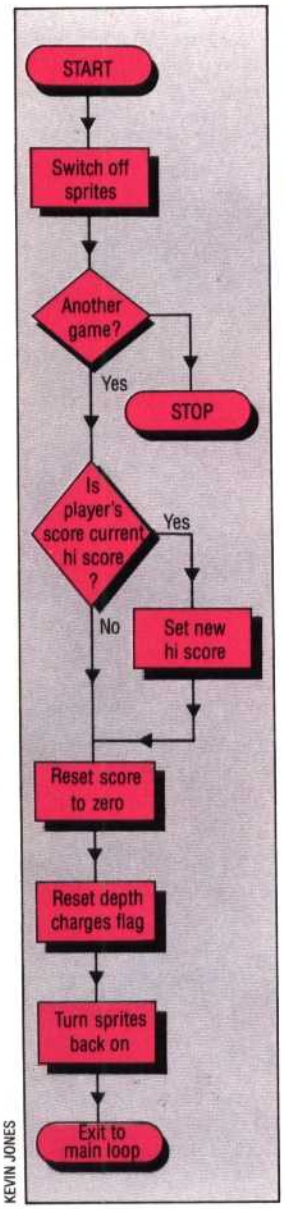
RUN SILENT, RUN DEEP



KEVIN JONES

At last we can apply the finishing touches to our Subhunter game. We set up the routines that create an explosion when a depth charge hits a submarine, and explain the end of game procedure.

In the penultimate instalment of the course, we discovered how easy it is to detect collisions between sprites using a sprite collision register, V+30. When this happens, the Hit subroutine (starting at line 5000) has three tasks to perform. First of all, it must cause an explosion at the point on the screen where the two sprites collided, and then it must increase the player's score by the value of the sub, which is calculated from its speed (DX) and its depth (Y3). Finally, it must reset the co-ordinates for the next sub to start moving across the screen. Let's look at the code for the Hit subroutine (lines 5000–5250) in more detail.

Line 5010 POKEs a zero into the collision register V+30 to clear it. Commodore claims the sprite collision register clears itself once two sprites have passed over each other and are no longer in collision. Experience, however, shows that the register does not always clear itself quickly enough, causing unexpected effects such as explosions occurring for no reason. The solution is to clear the collision register manually after a collision. Once this has been done the explosion sprite can be positioned and turned on.

Line 5030 gives the explosion an X co-ordinate ten pixels to the right of that of the depth charge. This slight shift positions the explosion more centrally over the depth charges. As X2 takes its value from the ship's X co-ordinate (X0), its value has an upper limit of 245. This means that the maximum value of the explosion's X co-ordinate is 255. The Y co-ordinate for the explosion is taken directly from that of the submarine.

The explosion sprite has been designated as sprite 1. Line 5040 sets bit 1 of the register V+21 to one, turning on sprite 1 without disturbing the values of other bits within the register. At this point it is interesting to note that the explosion sprite will appear on top, or in front of, the sub and depth charge sprites. This is known as *sprite priority*, and it is governed by the simple rule that lower numbered sprites appear in front of higher numbered ones. It is no accident that the explosion was designated as sprite 1 and the depth charges and sub were designated 2 and 3 respectively.

The colour of the explosion sprite is controlled by location V+40 of the VIC chip. An interesting effect can be obtained by rapidly changing the colour of the explosion using a FOR... NEXT loop to POKE in colour code numbers between 1 and 15. An outer FOR... NEXT loop repeats this process 20 times (lines 5060–5100). When the explosion is complete, all three sprites (explosion, depth charges and submarine) must disappear from the screen. Line 5130 turns sprites 1, 2 and 3 off.

As mentioned previously, the player's score needs updating using the subroutine beginning at line 5500. As the score is to be increased by the sub's value (rather than decreased, as happens when a sub reaches the right hand side of the screen unscathed) the value of DS is set to one to signal this. Finally, before another sub can travel across the screen, its co-ordinates need to be reset using the subroutine at 2500 and the sub sprite must be turned back on. In addition, the flag that signals the dropping of a depth charge must be reset to zero so that the player can start firing depth charges again.

At the end of three minutes the program leaves the main loop and jumps to line 400. When we first discussed the use of the Commodore 64 timer (see page 234) line 400 was a simple END statement. The End of Game routine allows the game to be replayed and the highest scores recorded. The flowchart shows the tasks to be incorporated into such a routine. Lines 400 to 660 of the program listing perform these tasks. Most of the code is self-explanatory, remembering that CHR$(19) homes the cursor to the top left corner of the screen and CHR$(144) causes subsequent PRINTed letters to be coloured black.

In this short programming project for the Commodore 64 we have learned how to construct a simple animated game. In building up the program we have covered all the main aspects of programming this kind of game in BASIC. You may well wish to add refinements of your own to the program using the principles we have learned. One way of extending the game to make it more interesting would be to allow more activity on the screen by incorporating the four unused sprites.

A Table Of The Variables Used In Subhunter	
V	Start of the VIC chip registers
FL	Depth charges flag – set to one if a charge is dropped
SC	Current player's score
HS	Highest score so far
TI$	Commodore 64's own timer
X0	X co-ordinate of ship
X2,Y2	X and Y co-ordinates of depth charge
X3,Y3	X and Y co-ordinates of sub
H3,L3	Hi byte and lo byte of sub's X co-ordinate
DX	Number of pixels by which X co-ordinate of sub is increased
DS	Flags whether a score is to be increased (DS=1) or decreased
	(DS=-1)