Micro Processes

Keyboard Out Joystick In

by Michael Quigley

Many games written for the VIC 20, and a few for the Commodore 64, use keys such as I, J, L, and M (up, left, right and down), rather than a joystick, to move objects around on the screen. I've always found this curious, since just about everyone I know has a joystick.

Converting keyboard programs to joystick is not particularly difficult — you just have to know what to look for. Most of these programs use memory location 197 ('current key pressed') to direct the cursor, rocket ship or whatever, around the screen. Each key on the keyboard (except the logo key, the **SHIFT** keys, **CTRL** and **RESTORE**) has a value in this location when it is pressed. Type in the following two lines, type **RUN** and then press some keys.

```
10 print peek (197)
20 goto 10
```

Try typing I, J, L and M, and note the values. They should be 12, 20, 21 and 36 respectively on the VIC, and 33, 34, 42 and 36 for the C-64. Note that when no key is pressed, the value is 64 on both machines.

Now comes the detective work. Look through your program and try to find lines which are something like the following:

```
100 if peek(197)=64 then.. [no input]
110 if peek(197)=12 then.. [up]
120 if peek(197)=20 then.. [left]
130 if peek(197)=21 then.. [right]
140 if peek(197)=36 then.. [down]
```

Since the results of pressing these keys vary from program to program, the four example lines here are purposely left incomplete. The first line, corresponding to 'no key pressed', may or may not be present. Of course, some programs also allow movement in only two directions, not four. In such cases, the lines for horizontal or for vertical movement may also be missing. Another thing to watch for is if **PEEK(197)** is converted earlier in the program to a numeric variable, for example: **P = PEEK(197)**. Then the lines above would read **100 IF P = 64 THEN...**'', and so on.

Now that you've found the place where keyboard inputs are interpreted, what is the magic joystick formula to use? It's as follows:

```
0 goto 10: rem vic 20 version
1 qq=37154:q1=37151:q2=37152
2 poke qq, 127:q=peek(q2)and128
3 r0=-(q=0)
4 pokeqq,255:q=peek(q1)
5 d0=-((qand8)=0)
6 l0=-((qand16)=0)
7 u0=-((qand4)=0)
8 fb=-((qand32)=0)
9 return

0 goto 10: rem commodore 64 version
1 q=peek(56320)
2 r0=((qand8)=0)
3 d0=((qand2)=0)
4 l0=((qand4)=0)
5 u0=((qand1)=0)
6 fb=((qand16)=0)
7 return
```

This standard routine should be placed somewhere in your program, preferably at the beginning to speed things up. The line numbers can be changed if you wish. Notice that I have used some variable names that suggest the direction they will take — *U0* for 'up', *D0* for 'down', and so on. Be sure that none of the variables in the joystick formula are the same as those already in your program, otherwise things will get really messed up!

Now go back to those lines you found earlier — the ones for interpreting keyboard input. Make the following changes:

```
100 gosub 1:if u0 or l0 or d0 or r0 o
    r fb then 110
105 rem perform action for 'no input'
110 if u0 then... [up]
120 if l0 then... [left]
130 if r0 then... [right]
140 if d0 then... [down]
```

The new line 100 first **GOSUB**s to the joystick subroutine. If a joystick input is made, it jumps to line 110, where the input is decoded and acted upon. If *nothing* happens (the equivalent of **PEEK(197) = 64**), then the 'no key pressed' routine, if any, may be accessed in line 105.

In the last couple of examples, we've used a new variable, *FB*, for 'fire button'. This input may not be needed, but if it is, the process for making the conversion is analogous to the cases we have already discussed.

Another way in which keyboard inputs are handled on the VIC and C-64 is with a **GET** statement. Assuming we are using 'S' for fire, 'I' for up, 'J' for left, 'L' for right, and 'M' for down, a typical keyboard checking routine might look like this:

```
100 get q$
110 if q$="s" then 1000 [go to fire b
    utton routine]
120 if q$<>"i"andq$<>"j"andq$<>"l"and
    q$<>"m"then300 [no input]
130 x=x+(q$="j")-(q$="l") [determine h
    orizontal direction]
140 y=y+(q$="i")-(q$="m") [determine
    <space>vertical direction]
```

Here's the same routine, converted for joystick operation:

```
100 gosub 1 [go to joystick formula]
110 if fb then 1000 [go to fire butto
    n routine]
120 if u0+l0+r0+d0=0 then 300 [no inp
    ut]
130 x=x+(l0)-(r0) [determine horizont
    al direction]
140 y=y+(u0)-(d0) [determine vertical
    <space>direction]
```

I find using a joystick highly preferable to groping around the keyboard, and if you're really lazy, there's yet another trick to be learned. Instead of the usual keyboard input to rerun a program:

```
100 print"play again? (y/n)"
110 get a$:if a$<>"y"and a$<>"n"then1
    10
120 if a$="n"then end
130 if a$="y"then run
```

try the following:

```
99 rem vic 20
100 print "play again? press fire but
    ton"
110 wait 37137,32
120 wait 37137,32,32
130 run
```

```
99 rem commodore 64, port #1
100 print "play again? press fire but
    ton"
110 wait 145,16
120 wait 145,16,16
130 run
```

```
99 rem commodore 64, port #2
100 print "play again? press fire but
    ton"
110 wait 56464,16
120 wait 56464,16,16
130 run
```

With this routine, your hands need never leave your joystick, except when you want to stop the game with **RUN/STOP-RESTORE**! □