The Variable-Duty-Cycle Algorithm

Timothy Stryker, Software Technology Inc, Precision Rd
Danbury CT 06810

Every now and then a novel technique for handling data comes to light which, while not immediately obvious, is actually very simple and can be used in a wide variety of applications.

One example of such a technique is the use of the *linked-list* data structure, which allows the programmer to create ordered sets of entries into which new entries can be inserted and from which existing entries can be easily deleted. Another example is the use of *semaphores*, which have many applications that center around the allocation of resources to sets of processes.

A new technique in this category has recently been added to the list. Pioneered by Albert G Love of General DataComm Industries, the technique initiates an event so that it occurs a specified proportion of the times that another event occurs.

Let us call the event that is conditionally initiated the "kickee," and the other event, typically something that occurs at even intervals in time or space, the "kickor." Now define three quantities, called the *duty-master*, the *duty-cycle*, and the *duty-counter*. The ratio of the duty-cycle to the duty-master will determine the proportion of kickee to kickor events. (The duty-counter is a scratch quantity that will ordinarily be initialized, say at power-up, to zero.) Each time the kickor event occurs, we do the following:

```
duty-counter := duty-counter minus duty-cycle
if duty-counter is now negative
then do
    <initiate kickee>
    duty-counter := duty-counter plus duty-master
end
```

This procedure may seem sufficiently abstract as to be totally useless, so let's consider a concrete example: a D/A (digital-to-analog) converter constructed of one bit from a computer parallel output port, one resistor, and one capacitor. The resistor and capacitor are connected so as to form a simple low-pass filter, as shown in figure 1. Now you can run the BASIC program shown in listing 1 on the computer.

Depending on the values of resistance and capacitance, and the speed at which the program executes, the voltage at the analog output point will be a more or less steady 3.75 V. By changing the constant in the DATA statement in line 100, any arbitrary voltage between 0 and 5 V can be obtained.

In this example, the duty-master is the constant 5 appearing in line 70, the duty-cycle is the variable V, and the duty-counter is the variable C. The kickor is the occurrence of a pass through the loop extending from line 30 to line 90, and the kickee is the decision to output a 1,

Listing 1: *This BASIC program uses the VDC algorithm to provide a steady output voltage when combined with the simple circuit in figure 1. A change in the value of the DATA statement will alter this voltage. Program line 80 must output the contents of B to the appropriate output port.*

```
10 C=0
20 READ V
30 B=0
40 C=C-V
50 IF C>=0 THEN 80
60 B=1
70 C=C+5
80 <output B to port>
90 GOTO 30
100 DATA 3.75
```

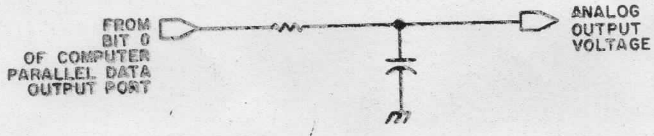

Figure 1: *Low-pass filter that converts the digital output of the single-bit data port to an analog signal.*

instead of a 0, to the port. The utility of this example could be considerably enhanced through the use of assembly language and real-time interrupts, but the utility of the basic scheme should be clear: assuming that each pass through the loop requires the same amount of time, the waveform output to the port will have an average duty-cycle precisely equal to the ratio between the duty-cycle, V, and the duty-master, 5. In addition, the waveform will bounce back and forth between 0 and 5 V at the maximum possible rate given the desired duty cycle and the available processing time, which will make the low-pass filter's job as easy as possible in reducing ripple at the analog output.

The variety of ways in which this same basic technique can be applied is extraordinary. Consider the case in which two integer quantities need to be kept as close to a given ratio as possible while both are gradually increased from zero to some higher number. Normally this would involve substantial amounts of multiplication and/or division, or have drawbacks in terms of either minimum increment size or worst-case error. However, use of the VDC (Variable-Duty-Cycle) algorithm makes the task straightforward: simply call the two integers I and J, and let the desired ratio between them be K:L. Pick a number, M, which is greater than or equal to both K and L, and, each time you wish to increase I and/or J by a small amount, do the following:

```
C := C-K
if C<0
then do
    <increment I>
    C := C+M
end
D := D-L
if D<0
then do
    <increment J>
    D := D+M
end
```

This process, of course, merely combines two instances of the VDC algorithm, using a common duty-master, M. The duty-cycle quantities are K and L, the duty-counters are C and D. The method requires virtually no processing time or memory space, is completely processor- and language-independent, and presents no theoretical limitation on the degree of precision with which the desired ratio may be maintained.

Technical Forum is a feature intended as an interactive dialog on the technology of personal computing. The subject matter is open-ended, and the intent is to foster discussion and communication among readers of BYTE. We ask that all correspondents supply their full names and addresses to be printed with their commentaries. We also ask that correspondents supply their telephone numbers, which will not be printed.

Nothing that we have seen so far suggests that the duty-cycle quantity could not vary from one occurrence of the kickor to the next. This is very handy for, among other things, modeling the effects of acceleration and velocity upon the position of an object. Suppose we are designing a real-time graphics game in which there will be a cannon capable of launching a projectile on a parabolic path toward a target. Is it possible to generate a parabolic path without resorting to a multiplication routine? Indeed it is!

We accomplish this by treating the projectile's horizontal and vertical velocity components as duty cycles, where the common kickor is a routine that runs at evenly spaced intervals in time, and the kickees are routines that move the projectile one cell horizontally and one cell vertically. Typically, the projectile's horizontal velocity component is a constant in the forward direction, and is easy to handle using the formula we have seen here. To deal with the possibility that the projectile could, vertically, be moving either up or down, we will have to introduce the concept of a negative duty-cycle. If M is the duty-master, H and V the horizontal and vertical duty-cycle/velocities, and C and D the duty-counters, the kickor routine looks like this:

```
    C := C-H
    if C<0
    then do
       <move projectile one cell to the right>
       C := C+M
    end
    D := D-V
    if D<0
    then do
       <move projectile one cell up>
       D := C+M
    end
  else if D> =M
  then do
     <move projectile one cell down>
     D := D-M
  end
<decrease V by a fixed amount>
```

Photo 1 shows the set of projectile positions that are obtained when M is 125, H is 25, and V starts off at 75 and is decremented each kickor pass by 1 until it reaches −75.

More complex (but perhaps less useful) patterns can be generated if the kickee is permitted to change the value of the duty-cycle or duty-master, if the kickee of one VDC is made the kickor of another, and so on. But even in the simple forms given here, the applications of this algorithm range from data-communications multiplexing to printer/plotter control to industrial process simulation to—who knows what? Perhaps you will be the next to add to the list. ■

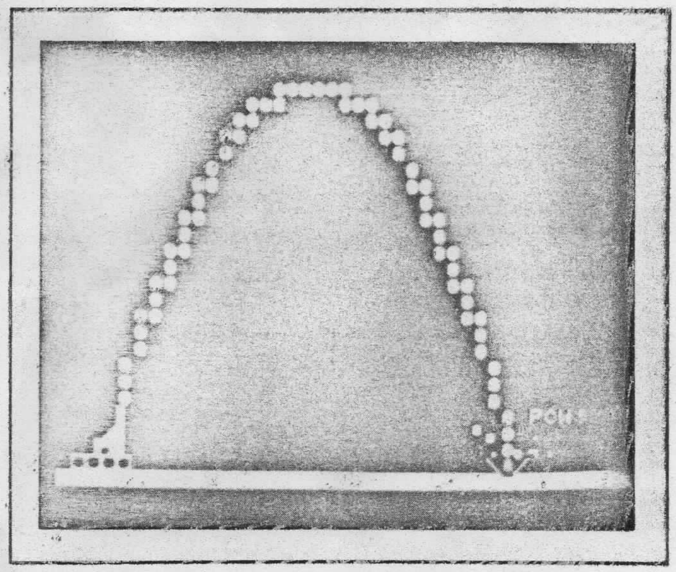

Photo 1: *The VDC algorithm can be used to predict the parabolic path of a projectile without the use of multiplication or division.*