Almost all graphics calculators are programmable, although the mathematics education community seems not to have made extensive use of this feature yet. A possible reason for this neglect is the widespread view that programming is difficult, or at least complicated, and perhaps a concern that not much is to be gained by learning the new languages required.

In fact, such views are mistaken. The programming languages of graphics calculators are often easy to use, and there are substantial mathematical benefits to be had from using them. A further discussion of this view and some examples of short programs for Sharp EL-9300 and TI-82 graphics calculators are given elsewhere (Kissane, 1994a; 1994b; 1995). However, in this paper, the intention is to illustrate the ease with which programs occupying only a single command line of a graphics calculator can be constructed, and to suggest some kinds of educational and mathematical uses to which this capability might be put. In most cases, what I have come to describe as ‘one-line programs’ require little knowledge of programming beyond an awareness of how a particular calculator works.

The least sophisticated example of a one-line program is arguably not on a graphics calculator at all, but on a four-function calculator. On many of these, after the display is cleared, a command like \[ \text{...}\] will have the effect of displaying the counting numbers in succession each time the equals key is pressed. The effect of this ‘program’ is to turn a calculator into a counting machine, a transformation put to great use by many teachers of very young children, delightfully illustrated on the video, *Young children using calculators* (Deakin University, 1994). (On some models of calculators, the appropriate key sequence is \[ \text{...}\], which has an extra plus sign.) Each press of the equals sign has the effect of repeating the previous command, consisting only of the instruction to add one. One-line programs like this can be constructed and used to educational advantage with subtraction and multiplication as well.

**Basic examples**

Similar commands to those on a four-function calculator will also work on a graphics calculator, since pressing the (equivalent of the) equals key on a graphics calculator will either execute the present command line or, if none has been entered, execute the previous command line again. This inbuilt operation of repeating a command so easily is what gives the graphics calculator the capacity for powerful one-line programming. To illustrate, the screens below, like others in this paper, are taken from a Casio fx-9700 graphics calculator. The number 5 was first entered. Then the one-line program

\[ \text{Ans} + 4 \]

was entered, just by pressing \[\text{+}4\]. As the screen shows, repeating this command, by pressing the \[\text{EXE}\] key, gives successive terms of an arithmetic sequence.
The same idea works nicely with multiplication, generating successive terms of a geometric sequence. For example, the screens below show the growth of an investment of $500 with an interest rate of 7.5%, compounded annually. After four years, the amount has accumulated to $667.73:

Students can readily explore growth and decay situations with a simple one-line program like this.

Nor is it only sequences that can be generated with one-line programs like these. Most graphics calculators include a pseudo-random number generator, so that a stochastic process can readily be simulated. The screen below shows a command to convert a random number on the fx-9700, originally in the interval (0,1), to an integer in the set \{1,2,3,4,5,6\}. This has the effect of simulating a toss of a fair die. Six successive tosses are readily simulated by this one-line program by pressing the \texttt{EXE} key six times, thus repeating the same command each time.

A more complicated command allows tossing a pair of dice to be simulated:

The ease with which such experiments can be performed provides a fresh opportunity for students, individually or collectively, to explore random situations informally. More sophisticated students could even examine the properties of the calculator's random number generator with data obtained in this way.

**More complicated recursion**

Many recursively defined sequences cannot be readily produced with a single arithmetic operation. An interesting example is the well-known logistic formula, given by the recursive relationship,

$$X_{n+1} = AX_n(1 - X_n).$$
With a suitable starting value $X_0$ in the interval $(0,1)$ and a value for $A$ in the interval $(3,4)$, all sorts of interesting things happen, some of them leading to the fascinating ideas of chaos.

One way to produce successive terms of such a sequence using a one-line program is to use the calculator variable $Ans$, used by the Casio fx-9700 (and many other models of graphics calculator) to store the results of the most recent calculation. The screen below shows the sequence with $X_0 = 0.4$ and $A = 3.1$.

\[
\begin{array}{c}
0.4 \\
3.1\text{Ans}\langle 1-\text{Ans}\rangle \\
0.744 \\
0.5904384 \\
0.749644776397 \\
0.581800294486
\end{array}
\]

The initial value has been entered first, followed by the recursive one-line program. Then each successive press of the $\text{EXE}$ key produces another term of the sequence. For example, $3.1(0.4)(1 – 0.4) = 0.744$ and $3.1(0.744)(1 – 0.744) = 0.5904384$.

Precisely the same idea can be employed without using the $Ans$ variable, but using one of the calculator’s memories, and updating the memory by storing the new result back into the same memory each time. The screen below shows this, using the variable $X$ and with an arrow representing the memory storage operation:

\[
\begin{array}{c}
0.4 \times X \\
3.1 \times (1 - X) \rightarrow X \\
0.4 \\
0.744 \\
0.5904384 \\
0.749644776397 \\
0.581800294486
\end{array}
\]

With this kind of one-line program, students can explore discrete dynamical systems and elementary ideas of chaos with comparative ease. Once the basic idea is known, many recursive relationships can be explored in a similar way.

An especially useful recursive operation is the Newton-Raphson procedure for finding numerical approximations to the roots of a function $F(x)$:

\[x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)} \, .\]

With a suitable choice of starting value $x_0$, this iteration allows for the roots of any elementary function to be approximated. Thus, finding the root of $F(x) = x^3 - x - 1 = 0$ involves using the recursion

\[x_{n+1} = x_n - \frac{x_n^3 - x_n - 1}{3x_n^2 - 1} \, .\]

After starting with $x_0 = 1.2$, estimated from a graph, the screen below shows that this process converges quite quickly to the required root of $x \approx 1.3247$:
The educational significance of this kind of one-line program is that it provides an efficient mechanism to find numerical solutions to most elementary equations, even if students do not have an automatic equation solver on their calculator. Attention can then focus on setting up equations, interpreting their solutions, and examining convergence, rather than merely the mechanics of solving them.

The example above involves a function for which the derivative is known. A similar procedure can be used on the fx-9700 when the derivative is unknown, making use of the calculator’s numerical derivative command, shown as d/dx on the screen below.

The results for this one-line program also converge quickly.

**Multistatements**

Although much can be achieved with a one-line program consisting of a single statement, many modern graphics calculator allow for multistatements, referring to the stacking of several statements onto a single line. Generally, graphics calculators use a colon to separate consecutive statements on the same line. Each time the command line is entered (on the Casio fx-9700, by pressing [EXE]), all of the statements are executed in turn.

There are many uses of this facility to allow a one-line program to have more than one step. For example, as well as obtaining successive terms of a sequence, the corresponding series can be obtained by accumulating terms into another calculator memory. The screen below shows one way of doing this, in a one-line program dealing with the sum of the squares of the integers.

The first two commands initialise the term number (X) of the sequence and the partial sums (S) of the series. Then the one-line program replaces X with the next term number (X+1), adds its square into the partial sum (S) and then displays the new term (X²). The three commands are separated by colons, so the one-line program takes the form of a multistatement.

For students unfamiliar with programming, a command like \( X+1 \rightarrow X \) may appear strange at first. It is perhaps more accessible, however, than the equivalent statement \( X = X+1 \), used in some algebraic computer programming languages like BASIC, since such statements seem to be equations with no solution.
After a number of terms have been added, the usual memory recall command allows the value of S to be found. The screen below shows that the sum of the squares of the first 30 integers, after the required number of presses of \([\text{EXE}]\) is \(S = 9455\).

A multistatement can also be used to construct recursive sequences involving more than a single term, such as the Fibonacci Sequence. In the screen below, \(X\) and \(Y\) represent two successive terms, and variable \(Z\) is used to store (temporarily) the next term, given by \(X + Y\).

The first multistatement in this screen initialises the sequence. The next multistatement is a one-line program to produce the next term and update the last two terms.

Adding an additional step in the one-line program allows the ratio of successive terms to be printed each time \([\text{EXE}]\) is pressed:

With such a program, students can readily examine convergence. In this case, the ratio converges quite rapidly to the golden ratio, \(\phi = \frac{1 + \sqrt{5}}{2}\), as is well known.

**Program input**

A further embellishment of the idea of one-line programming involves allowing input statements to be made. Most calculators allow such statements to be made only in the programming area of the calculator, but the entire Casio range of graphics calculators has an input command that is available in computation mode as well as in programming mode. When used in conjunction with the multistatement facility, this command allows very useful one-line programs to be written.

For example, the one-line program shown below first obtains a value for the variable, \(X\). (The input command is shown as a question mark.) Then the program prints the value of the expression \(2X^2 - X + 1\). Each time \([\text{EXE}]\) is pressed, a fresh value of \(X\) is sought, and the corresponding value of the expression computed. This is a very handy way of printing out several successive values of the same function. It might also be useful for obtaining a quick ‘guess and check’ numerical solution to an equation, or for obtaining several points to help
sketch a curve by hand.

Not all formulas have one variable, however. But the same idea allows the Casio fx-9700 to readily evaluate functions of several variables, by using a multistatement that incorporates more than one input statement. An example is shown below, in which volumes of successive cylinders are obtained after values for the radius (r) and the height (H) are entered.

One-line programs of this kind can save both students and their teachers lots of time, as well as helping students to form a good idea of the concepts of a variable and of a formula.

**Program output**

All the examples of one-line programs so far have involved a single output. It is also possible to write one-line programs with multiple outputs, using a display pause command. On the Casio range, this command is represented by a small triangle. To illustrate how this works, the screen below shows a one-line program for finding both roots of the general quadratic equation \(Ax^2 + Bx + C = 0\).

This is a complicated one-line program (taking up *three* lines on the screen!) involving three input commands, a command to calculate the discriminant, \(D\), and two commands to output the two roots of the equation. Nonetheless, it is still a one-line program, since the calculator repeats the entire program each time it is initiated. The screen below shows the program in use to solve \(x^2 + x - 6 = 0\).

When the roots are complex, the Casio fx-9700 automatically uses the conventional complex number symbols to show this, as the following screen illustrates with the solution of \(x^2 + 2x + 2 = 0\).
Younger students who have not yet encountered complex numbers may be surprised at the appearance of the $i$ symbol in this way, but older and more sophisticated students and their teachers may find such a one-line program a useful alternative to the tedium of solving quadratic equations in other ways.

It may be, however, that the spirit of one-line programming is beginning to be compromised with single ‘lines’ that are as long as this one, and someone starting to write such programs may be well advised to use the programming area of the calculator instead. An advantage of doing so is that the programs can be stored for the longer term, and can be much more easily edited.

**Conclusion**

Graphics calculators are all too frequently regarded merely as devices for drawing and exploring graphs of functions. However, modern graphics calculators such as Casio’s fx-9700 have the potential for many other useful mathematical operations for students or their teachers. The idea of one-line programming exploits the inbuilt capability of the calculator to repeat the same command line, and thus to provide opportunities for exploring iteration, repetition and recursion. This paper has illustrated some of the ways in which this idea can be used to good effect in secondary school mathematics, without the need for students to be expert programmers.

**References**

Deakin University (1994) *Young children using calculators* (video), Geelong.


*A version of this paper was published as follows: