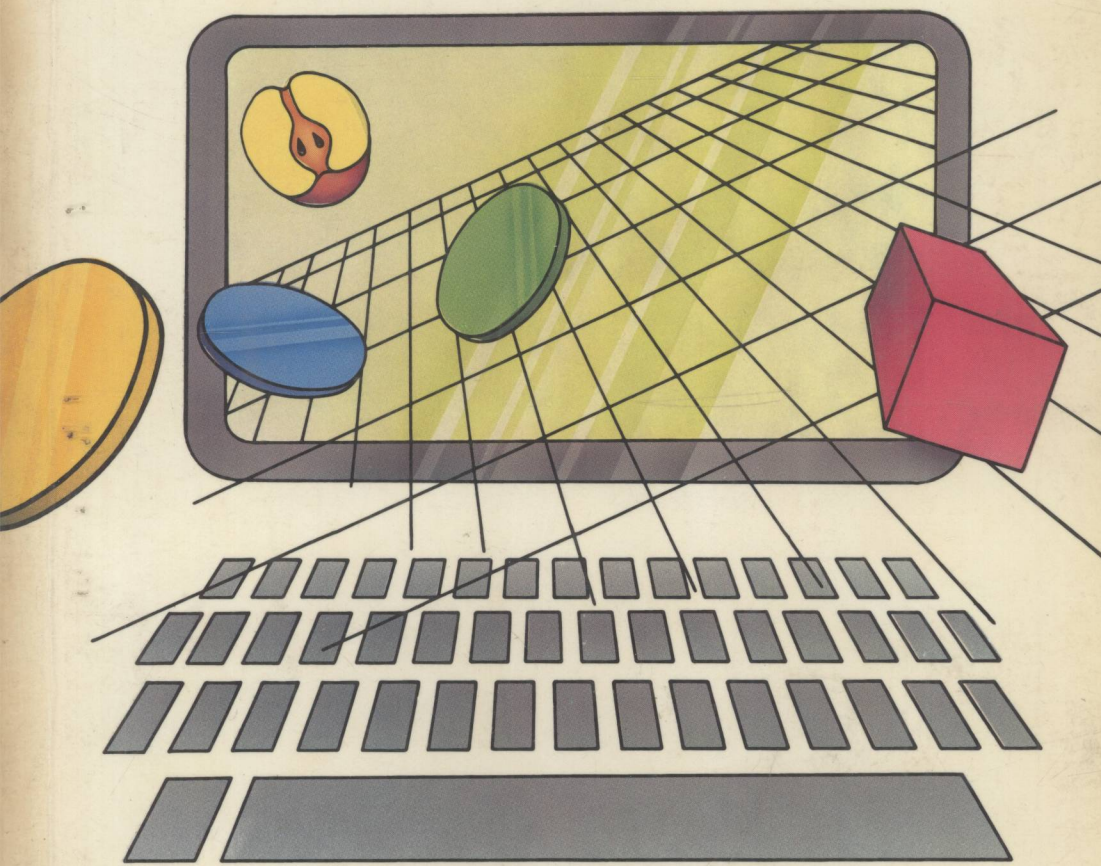
Exploring applesoft



ROGER McSHANE

EXPLORING APPLESOFT

Roger McShane

Prentice-Hall of Australia

A Prentice-Hall Direct Edition

Prentice-Hall of Australia Pty Ltd, Sydney
Prentice-Hall International Inc., London
Prentice-Hall Canada Inc., Toronto
Prentice-Hall of India Private Ltd, New Delhi
Prentice-Hall of Japan Inc., Tokyo
Prentice-Hall of Southeast Asia Pte Ltd, Singapore
Editora Prentice-Hall do Brasil LTDA., Rio de Janeiro
Whitehall Books Ltd, Wellington
Prentice-Hall Inc., Englewood Cliffs, New Jersey

1 2 3 4 5 87 86 85 84 83

Printed and bound in Australia by
Globe Press Pty Ltd, Brunswick, Victoria

Cover by Sam Latsis

ISBN 0-13-295916-X

Library of Congress Cataloguing in Publication Data

McShane, Roger, 1950-
Exploring Applesoft.

Includes index.
1. Apple II (Computer)--Programming. 2. Apple IIe (Computer)--Programming. I. Title.
Qa76.8.A662M4 1983 001.64'2 83-11152
ISBN 0-13-295916-X

National Library of Australia
Cataloguing-in-Publication Data

McShane, Roger.
Exploring APPLESOFT.

Includes index.
ISBN 0 7248 0417 X.

1. Apple II (Computer) - Programming.
I. Title

001.64'24

PREFACE

This book has been written in response to the needs of three groups: students having their first contact with a microcomputer, students who wish to study programming as an interest rather than a discipline, and last, but by no means least, adults who find the traditional approaches to programming to be difficult or confusing.

The main philosophy behind the approach taken in this book is that students will grasp programming concepts more readily if they can 'see' the results of their programs. For this reason there is a heavy emphasis on the use of graphics throughout the book. The author believes that students are able to detect errors, and correct those errors, much faster if they have a graphics image to deal with.

Another variation from the traditional approach is that the syntax of the APPLESOFT language is introduced incidentally. The author believes that many students are given a very bad impression of programming in their first few lessons when a bewildering number of 'rules' for PRINT formatting or mathematical formulae are thrust upon them. The time for an explanation of these facts is when the student needs to use them - if they are comfortable in using the computer for simple programming they will be more receptive to looking up the rules in books and manuals.

The book has been designed to appeal to a wide range of students and therefore does not contain many examples or exercises from the mathematics area. Computers are used widely in the general community for accounting, word processing, file handling, record keeping, information sharing and for recreation - very few people use them for mathematics in the true sense and the author believes that the teaching of programming should reflect this.

In the first sections of the book a number of 'programming models' have been introduced to help explain some of the elementary programming concepts which are so often dealt with in a cursory fashion. It is the author's belief that students begin writing programs by imitating sections of code that they have already seen. It is not until they

have internalized a complete set of 'models' that they become proficient at the task of coding. One of the chief responsibilities of the teacher, therefore, is to provide students with an adequate set of models. Conversely, students should not be asked to attempt problems for which adequate models have not been provided.

The book has been divided into a number of lessons each with a particular theme, and these are supplemented by a number of 'interludes' and appendices which contain much of the technical information the students will need. Most of the lessons finish with a slightly longer program. It will not be necessary for all students to understand those programs as the next lesson will begin at an easier level in each case. The author has included these harder examples for those students who find programming to be easy. Towards the end of the book a number of programming standards are introduced to show students who intend to take their study of programming further that adherence to standards and programming style is essential in the programming field.

I would like to pay a tribute to my wife Ann and my children Stuart and Sallyanne who have supported me in the development of this book, giving up many weekends and holidays so that the project could be completed. My sincere thanks are also extended to Scott Brownell who has been an inspiration to all computer educators in Tasmania, and to Jo Ginn for the diagrams used in this book.

Roger McShane

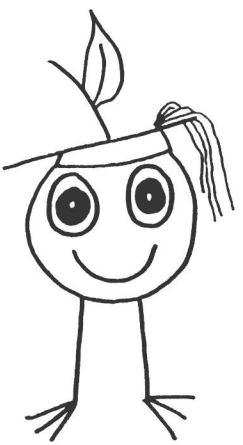
Hobart, 1983.

CONTENTS

Lesson 1

Graphics Programming

In this lesson the fundamental concepts of computer programming will be discussed. These ideas will be introduced by plotting various images on the Apple's graphics display. The fundamental ideas of assigning values to variables and constructing loops and branches are introduced.

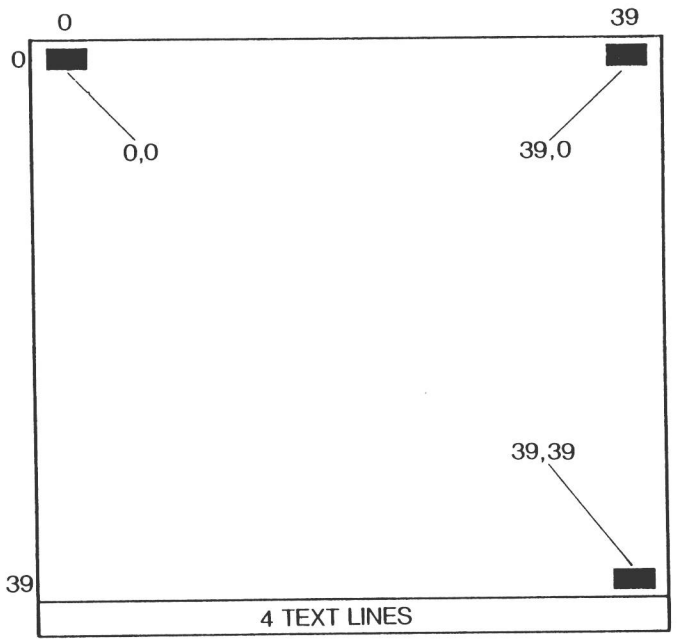


Our investigation of the way to program the Apple microcomputer will begin with a look at the GRAPHICS features. The Apple must first be turned on. Throughout these lessons it will be assumed that the Apple has been turned on and Applesoft is the standard language. The prompt] will appear as soon as Applesoft is available.

The Apple can display TEXT (ordinary characters) or GRAPHICS (pictures and designs) or a mixture of both. One of the graphics screens (the low-resolution) will be displayed in response to the instruction:

```
] GR
```

All of the screen, except for the bottom four lines, will be cleared to prepare for the graphics display. The graphics display consists of 1600 points on the screen made up of 40 columns (numbered 0 to 39) and 40 rows (also numbered 0 to 39). The top left hand corner is numbered 0,0.



LOW-RESOLUTION GRAPHICS DISPLAY

Graphics points are displayed using the PLOT command. Try the instruction:

```
] PLOT 5,30
```

This instruction asks the Apple to display a point in the 5th column and the 30th row. After we have typed this instruction, however, nothing will be seen! The reason for this is that the Apple must know what COLOR to plot the point in. We have just plotted a black point on a black background.

The COLOR can be specified by typing

```
] COLOR=2
```

If we now repeat the instruction

```
] PLOT 5,30
```

a small, colored rectangle will appear on the left-hand side of the screen towards the bottom. The color codes are:

0..... BLACK	8..... BROWN
1..... MAGENTA	9..... ORANGE
2..... DARK BLUE	10..... GRAY
3..... PURPLE	11..... PINK
4..... DARK GREEN	12..... LIGHT GREEN
5..... GRAY	13..... YELLOW
6..... MEDIUM BLUE	14..... AQUA
7..... LIGHT BLUE	15..... WHITE

These colors will vary from one TV set or monitor to another.Try some different COLOR commands such as COLOR=9 or COLOR=12. Follow the COLOR commands with PLOT commands.

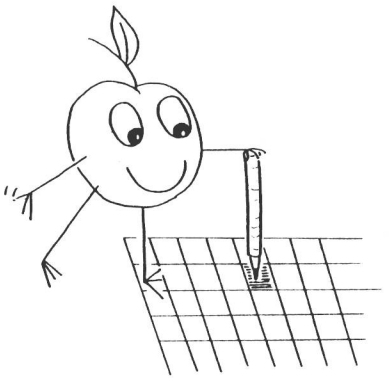
Throughout this book a number of PROGRAMMING MODELS will be provided to act as a guide for using various statements in your own programs.

. . . . . .

MODEL 1

A point is PLOTted in the current color according to the column and row which it is given.

PLOT 5,2



In the example given, the point will be plotted in the fifth column and the second row.

. . . . . .

Exercises

(a) Photostat the diagram of the low-resolution graphics display then mark in each of the PLOT commands.

(i)	PLOT 0,0	(ii)	PLOT 3,18	(iii)	PLOT 37,23
(iv)	PLOT 1,5	(v)	PLOT 17,15	(vi)	PLOT 39,20
(vii)	PLOT 20,39	(viii)	PLOT 0,5	(xi)	PLOT 0,39
(x)	PLOT 15,10	(xi)	PLOT 20,30	(xii)	PLOT 19,39

(b) Draw the outline of a yacht on a graphics sheet then write down the PLOT commands required to draw it.

(c) Teacher activity:

Pass out a graphics sheet to each student then call out a number of PLOT commands. For marking purposes pass around an overhead projector transparency with the correct positions marked on it. The student places this over his or her own answer to check.

. . . . . .

If we now wish to display a point then erase it, the following sequence of commands could be typed in:

```
] GR                  ..... display graphics screen
] COLOR=15            ..... set color to white
] PLOT 15,30          ..... plot a point in white
] COLOR=0             ..... set color to black
] PLOT 15,30          ..... point disappears
```

If we wish to plot a point, then erase it, then plot it at the next position and hence move the point across the screen it would be very tedious to type these commands in. This is because we are giving our commands to the Apple in what is known as IMMEDIATE MODE. An alternative method is to STORE a sequence of instructions (known as a PROGRAM) and then ask the computer to execute these instructions. A program to display a point would look like this:

```
] 10 GR
] 20 COLOR=4
] 30 PLOT 15,30
] 40 END
```

After typing these instructions you will notice that nothing has been displayed. This is because we have used line numbers (10, 20 etc.) to store the sequence of instructions (program) in the Apple's memory. To activate the sequence of instructions type the following:

```
] RUN
```

The point will now be displayed. If RUN is typed again the point will be displayed again and so on. To erase this program and start a NEW one type:

```
] NEW
```

Now you will notice that the program "disappears" underneath the graphics screen. When a program is being typed it is a good idea to return to TEXT mode by typing:

```
] TEXT
```

At this stage you will notice that the display looks very strange. If you wish to remove this, press RETURN a number of times.

The following program will display an orange dot in each corner of the screen:

```
] 10 GR              ..... display graphics
] 20 COLOR=9         ..... set orange
] 30 PLOT 0,0        ..... top left
] 40 PLOT 39,0       ..... top right
] 50 PLOT 39,39      ..... bottom right
] 60 PLOT 0,39       ..... bottom left
```

To activate this set of instructions remember to type:

```
] RUN
```

Notice that it is not necessary to reset the COLOR after each PLOT command.

At the end of this lesson there is a program which causes a "ball" to bounce off the edge of the screen. What are the instructions needed to do this? Let us first look at the set of instructions which display a point and then erase it:

```
] 10 GR
] 20 COLOR=15
] 30 PLOT 15,30
] 40 COLOR=0
] 50 PLOT 15,30
] 60 END
```

Now type:

```
] RUN
```

You will notice that the point hardly appeared. This is because the Apple executes the instructions very quickly. If we wish to slow the Apple down a bit, a good idea is to "waste" some time in between instructions. One way of doing this is to make the APPLE "count" to a large number such as 100 or 500 or 1000. The Apple does this with the following "magic" command (which shall be explained later):

```
] FOR I = 1 TO 100 : NEXT I
```

Therefore we could amend the previous program by typing

```
] 35 FOR I = 1 TO 100 : NEXT I
```

This would have the effect of inserting the counting instruction between lines 30 and 40. Now when RUN is typed a white dot will be plotted and will remain while the Apple counts to 100. Increase this value to 300 or 500 if you wish the dot to stay there longer.

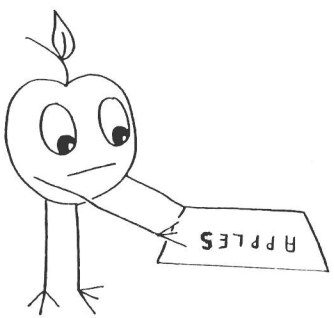
Let us now write a program to move the point across the screen at a rate we can see. Before this is done the concept of a VARIABLE must be understood. In the memory of any computer there are a large number of memory "cells" (variables) set aside to store numbers of characters. Study the next four programming models carefully before going on to the next program.

. . . . . .

MODEL 2

A memory cell or VARIABLE should be INITIALIZED (i.e. given a first value) before being used. The basket of apples will be used to represent the variable called APPLES. Some microcomputers, including the Apple, set all variables to 0 when RUN is typed. It is good programming practice, however, to give them an initial value. Notice that the cell has a name (LABEL) as well as a VALUE.

```
100 APPLES=0  .....          put 0 in the cell
                             labelled APPLES
```

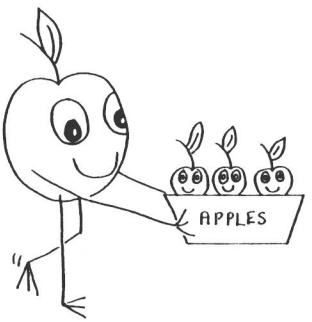


. . . . . .

MODEL 3

Numbers can be stored in VARIABLES. We say that the VALUE of 3 has been assigned to the VARIABLE called APPLES. Notice that there are three apples in the basket.

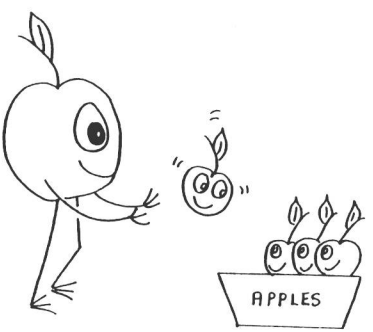
```
200 APPLES=3
```



MODEL 4

The value of a variable can be increased or decreased. In this example we say that "the variable APPLES is assigned the old value of the variable plus 1". In this case there were three apples in the basket, now there are four.

```
30 APPLES = APPLES + 1
```



. . . . . .

MODEL 5

A set of instructions can be executed over and over again through the use of a LOOP. Normally a program is executed line by line - the GOTO statement allows a change to a different part of the program.

90 GOTO 30 ..... go back to line 30

. . . . . .

The program to move a point across the screen can now be developed :

```
] 10 GR                          ..... turn graphics on
] 20 COLUMN=0                    ..... set column value
] 30 COLOR=15                    ..... set color value
] 40 PLOT COLUMN, 20             ..... plot first point
] 50 FOR I = 1 TO 100:NEXT I     ..... wait a while
] 60 COLOR=0                     ..... set color to black
] 70 PLOT COLUMN, 20             ..... erase point
] 80 COLUMN=COLUMN + 1           ..... move to next column
] 90 GOTO 30                     ..... do it again
```

When execution reaches line 90 it will simply jump back up to line 30 and start there again. When the command RUN is

issued, the white dot will move across the screen and then the message:

? ILLEGAL QUANTITY ERROR IN 40

will be displayed. The reason for this is that the variable COLUMN has taken the values 0,1,2,3, ... and so on, and has finally reached the value 40. There is no value 40, however, so the Apple has become confused! After the set of exercises, a method of testing the value of variables will be introduced. The test will help eliminate this error.

. . . . . .

Exercises

(a) Make the dot move across the screen very slowly.

(b) Make the dot move across the top of the screen.

(c) Make the dot move across the bottom of the screen. Note: it is not necessary to retype the whole program, just retype the lines which need changing.

(d) Change the program so that the dot moves down the screen in the center. Hint: change the variable from COLUMN to ROW and change the PLOT command to PLOT 20,ROW.

(e) Make the dot move down the screen on the left hand side.

(f) Make the dot change color each time it is plotted.

(g) Make the dot move across the screen, but this time only plot it every second point.

(h) Make the dot move diagonally from the top left hand corner to the bottom right hand corner. Increase a COLUMN variable and a ROW variable.

(i) Make the dot move from the top right hand corner to the bottom left hand corner.

. . . . . .

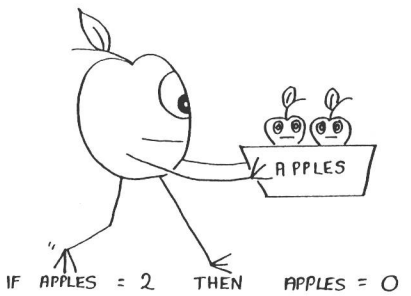
. . . . . .

MODEL 6

The value of a variable can be TESTED. If it passes the test then one set of instructions will be executed, otherwise a different set will be executed.

Example 1:

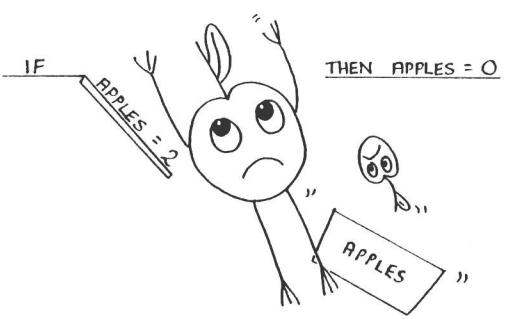
```
10 APPLES = 2
20 IF APPLES = 2 THEN APPLES = 0
```



In this example the value of the variable APPLES is initially 2 therefore the instruction after the THEN statement will be executed. Therefore the value of APPLES is now 0.

Example 2:

```
10 APPLES = 1
20 IF APPLES = 2 THEN APPLES = 0
```



In this example the value of APPLES is not 2 so the instructions after the THEN statement will be ignored.

. . . . . .