Programming by Experiment & Reverse Engineering
BBC BASIC (BB4W)

A Simple Solar System Simulation

The following exercise comprises of questions relating to a short program and experiments in changing the code.

In a previous exercise the author noted that all pupils should be able to work around sections of code that contain complex instructions. The argument for this, is that in everyday life we all solve problems that include extraneous information...

By not limiting the code, we are exposing pupils to the broader reality of programming and problem solving, and increasing the scope for experimentation. Questions and experiments can still be targeted on the simpler aspects of a program, or language, specifically for beginners.
The ability to filter out unwanted information, or focus on relevant information, is an important skill relating to decomposing, computational thinking and general problem solving.

The example code used below is not simple! However, bearing in mind the above, the questions and experiments included below could be a class exercise or could be completed by pupils working independently.

```
 1 REM Simple Solar System Sim
 2 MODE 9:REM Set the display mode
 3 OFF:REM Turn the text cursor off (invisible)
 4 ORIGIN 640,512:REM Move the graphics origin to the centre of the window
 5 *FONT Arial,12,b
 6 DIM slr(8,2)
 7 DIM p$(8)
 8 FOR a=0 TO 8
 9   READ p$(a),slr(a,0),slr(a,1),slr(a,2)
10 NEXT
11 p=3
12 MAXr=400
13 day=0
14 ON MOUSE:p+=1:p=p MOD 9:RETURN
15 REPEAT
16   CLS:REM Clear the screen/display
17   PRINTTAB(0,0);"Click mouse to move outwards"
18   day=day+slr(p,1) DIV 720
19   PRINTTAB(0,2);"Day: ";day DIV 10
20   GCOL 0,3:REM Select graphics colour Yellow
21   CIRCLEFILL 0,0,8
22   FOR a=0 TO p
23     r=slr(a,0)*MAXr/slr(p,0)
24     angle=360*(day MOD slr(a,1))/slr(a,1)
25     x=SIN(RAD(angle))*r
26     y=COS(RAD(angle))*r
27     GCOL 0,slr(a,2)
28     PLOT x,y
29     IF a>p-4 THEN
30       VDU 5:REM Special command to print text at graphics cursor
31       MOVE x+8,y-8
32       PRINT p$(a)
33       VDU 4:REM Special command to print text at text cursor
34     ENDIF
35   NEXT
36   WAIT 4
37 UNTIL day=-1
38 END
39 REM Name, Mean dist from sun in millions of km, Period of revolution in tenths of days, Colour
40 DATA Mercury,57.9,879,1
41 DATA Venus,108.2,2247,7
42 DATA Earth,149.6,3653,2
43 DATA Mars,227.9,6870,1
44 DATA Jupiter,778.3,43325,7
45 DATA Saturn,1427,107676,3
46 DATA Uranus,2871,306850,6
47 DATA Neptune,4497,602745,6
48 DATA Pluto,5913,905944,7
```

Questions:

1. What is the first line number containing data for planets?
2. Planet data are read into 2 arrays. On which lines are these arrays defined?
3. How many lines of data will be READ by the code?
4. Which line contains the instruction indicating the END of the program?
5. How many days will the simulation actually continue for, if uninterrupted?
6. On which text line will the current day appear?
7. Which line draws the sun?
8. Which line selects the colour to plot the planet?
9. Can you guess which line intructs the computer to react to user input?

Experiments:

1. What happens if you change line 36 to 'WAIT 20'?
   Change it back to WAIT 4 afterwards!
2. Can you change the colour of the sun to red (1)?
3. Can you change the colour of pluto to pink (5)?
4. Can you rename Mercury to display your name instead?
5. Add the following line to make a ring around Saturn:
   IF a=5 THEN CIRCLE x,y,12
6. What is the effect of changing line 20 to:
   GCOL 0,RND(7)
7. Can you make the sun bigger?
8. Change line 19 to:
   PRINTTAB(40,0);day DIV 10
   Note: ';' is a semi-colon!
9. Remove line 16 and see what happens.

Answers, with extended explanations:

1. Line 40

2. Lines 6 and 7. The instruction 'DIM' dimensions (creates) an array.
   slr() is a two dimensional array, which is a table or can be thought of as the cells on the page of a spreadsheet.
   p$() is a list of planet names. A list only has one dimension. In BBC BASIC, the $ in its name tells the computer that this array contains text, not numbers.
   It is possible for arrays to have 3 dimensions or more!

3. 9 DATA lines because the loop counts from zero to 8, not 1 to 8

4. Line 38, but it is never executed (see below).

5. Forever, because the variable 'day' will never equal -1

6. Text line 2, counting down from text line zero at the top of the display

7. Line 21. The sun is drawn as a filled circle

8. Line 27. The planet colour is defined in the DATA lines, which is READ into the slr() array

9. Line 14

   This is a very special line because it instructs the computer to execute the code on line 14 whenever a MOUSE button is clicked.
   It is also the only multi-statement line, with multiple instructions separated by colons ':'
   When the computer executes the code on this line, and eventually finds the RETURN instruction, the computer then returns to whatever it was doing previously.
   This type of code is described as 'event driven', because it only executes in response to a special event, and diverts from whatever it was doing at the time!