6 Diagrams and Data Graphs

More information is available to more people than ever before. Businessmen are being overwhelmed by massive documents that contain reams of statistics on every subject from capital expenditure to market research. Worst of all, computers are pouring out printouts of dreary data that cover every topic from Astrology to Zoology. Obviously something must be done! Computers have helped to create the problem and they can also help to solve it. The data must be presented in a more digestible manner: as pie-charts, histograms, scientific graphs or just plain diagrams. With the advent of desktop computers the increasing sales of programs that produce these displays has made this one of the major growth areas in computer graphics. In this chapter we shall see how such diagrams can be constructed with ease, given just a few tools to aid our draughtsmanship.

There are so many different types of diagrams that it is impossible to cover every possibility. We shall concentrate on the drawing of histograms, pie-charts and data graphs as well as giving a simple method for labelling and adjusting general types of diagram. Hence we give four major listings, which contain interactive programs for producing these diagrams, as examples of how to approach the general problem of data diagram construction. Naturally each individual program will require vastly differing data, although they will have some common input, display menus and prompts. Furthermore during the execution of a program the data being read may depend on previous responses (the program must not ask you for a radius before you ask to draw a circle!). Therefore we have organised the responses and displays (and the equivalent procedures) as part of a *question and answer* program that is based on the level concept. In our case level 0 will be common to all four programs, whereas lower levels will be unique to each of the four types of program.

To cope with the need for displaying both diagrams and prompts, the screen is divided into two areas: the graphics area which holds the diagram, and a text area for menus and prompts. In the top of the text areas we place five coloured blocks (normally logical colour 1 : default red) and marked f0, fl, f2, f3 and f4X, Y. Also, by pressing f9 when the machine requires input will cause a termination of the program. We call these the pseudo-soft keys because they correspond to the soft keys at the top of the keyboard. At any given moment during the execution of the program the machine can write character strings beneath each pseudo-soft key to demonstrate the option that is currently available with that key. Other prompts that request normal INPUT can also be printed in this text area beneath the blocks. When a soft key has been pressed then the colour of the equivalent pseudo-key turns yellow. Key f4 controls a crosswire cursor which, when activated, can be moved about the screen using cursor keys. However if you type f4 (when the corresponding pseudo-key is red) then a new position of the cursor can be explicitly INPUT as the coordinates of an addressable point. The present cursor position is written under the f4 pseudo-key.

The level 0 prompts consist of five options: SAVE (f9) to ‘save’ a picture on disk or tape, LOAD (f1) to re' load’ it from such a backing store, ERASE (f2) to erase the present picture (after double checking), the red f4 key for cursor control, and ETC. (f3). The latter option leads on to level 1, which naturally depends on which one of the four main programs has been loaded. The procedures for this level are given in listing 6.1 .

There are five procedures which ‘init' ialise various variables that are used by the procedures on all levels. The first, ‘initdims' , initialises any DIMensions that are required for any of the procedures. The ‘initkeys’ procedure sets up the variables that are needed for reading the soft keys and for the display of soft key prompts at the bottom of the screen. The procedure also uses the Command Line Interpreter (see the user guide) to clear any definitions off the first five soft keys so that no garbage is inadvertently entered into the programs. ‘initprompt’ sets up the strings that correspond to the red key prompts that are displayed in the text area. The fourth ‘init’ procedure, ‘initdiag' , sets up the graphics window that is used for drawing and clears the background to a colour, in this case logical 0. Finally ‘initcursor’ sets the crosswire cursor at its start position and initialises the variables that are used in the main cursor procedure.

Next we have a set of three procedures that can be used to display the set of simulated function keys. The ‘prompt’ procedure displays any one of a set of options for the key assignments: it uses the ‘text’ procedure to control the number of lines available at the bottom of the screen. A non-zero parameter for ‘text’ specifies the number of lines, and then it clears the screen, whereas a zero parameter sets the text window to cover the whole screen but does not clear the screen. The ‘light’ procedure lights up the pseudo-soft keys in specified back,round and foreground colours (defaulting to red, yellow or white).

Our construction of a ‘cursor’ requires a little further discussion. The need for accurately controlling the position of objects on the screen is self-evident. This is achieved on most graphics displays by a crosshair (crosswire) cursor, which may be controlled either from the keyboard or in more expensive devices from an external joystick, lightpen or similar analogue input device. Not wishing to put the reader to any further expense we shall use the keyboard to control movements: it achieves the same effect as a joystick anyway. The ‘cursor’ procedure that is contained in listing 6.1 overlays the existing picture with the crosswires by using the EOR option of GCOL. These crosswires specify the point at their intersection. The cursor is moved in any one of eight directions by the standard cursor keys either singly or in pairs. If you have a joystick or similar peripheral attached to your computer then alter the ‘cursor’ procedure so that it receives informittion from your device rather than from the keyboard.

The ‘cursor’ procedure may be called to initiate a single movement by using the parameter 1 in the call (externally initiated by pressing the f4 key when the equivalent pseudo-key shows red and explicitly typing the coordinates, as in the ‘sketch’ procedure), or a continuous sequence of moves using parameter 0 (externally initiated by pressing the cursor keys). The coordinate values of the present cursor position are requested by the program when the pseudo-key f4 shows (default) white, and they are entered by pressing soft key f4. When a cursor key is held down the speed of movement gradually increases (a cursor that always moves just one addressable point per key depression is tedious to use!). To aid in positioning the cursor there is a grid which is switched on and off by pressing G. If operative then it is automatically removed when you press f4 to enter a point.

Listing 6.1

```
  10 MODE 1 : VDU23,1,0;0;0;0;
  20 PROCinitdims
  30 PROCinitkeys
  40 PROCinitprompt
  50 PROCinitdiag(0)
  60 PROCinitcursor(640,512)
  70 LEVEL=0 : OL=-1
  80 BC=0 : OC=1 : TC=2 : GC=3
 100 REM main loop
 110 REPEAT
 120 IF LEVEL<>OL THEN PROCprompt(LEVEL) : OL=LEVEL
 130 IF INKEY(K(0)) THEN PROCkey0
 140 IF INKEY(K(1)) THEN PROCkey1
 150 IF INKEY(K(2)) THEN PROCkey2
 160 IF INKEY(K(3)) THEN PROCkey3
 170 IF INKEY(K(4)) THEN PROCkey4
 180 UNTIL INKEY(-120)
 190 STOP

 200 REM initdiag
 210 DEF PROCinitdiag(BACK)
 219 REM set window and clear graphics to BACKground colour
 220 VDU24,0;65;1279;1023;
 230 GCOL0,128+BACK : CLG
 240 ENDPROC

 300 REM initkeys
 310 DEF PROCinitkeys
 320 DIM K$(4),K(4) : RESTORE 380
 329 REM remove key definitions from key 0 to key 4
 330 FOR I%=0 TO 4 : A$="KEY"+STR$(I%)
 340 $32512=A$ : X%=0 : Y%=127 : CALL &FFF7
 350 NEXT I%
 359 REM set up arrays of INKEY values and strings for display
 360 FOR I%=0 TO 4 : READ K$(I%),K(I%)
 370 NEXT I%
 380 DATA"  f0  ",-33,"  f1  ",-114,"  f2  ",-115,"  f3  ",-116
        ,"  f4 X,Y    ",-21
 390 ENDPROC
 400 REM prompt
 410 DEF PROCprompt(A)
 419 REM clear the bottom two lines
 420 PROCtext(2) : PROCtext(0)
 429 REM display keys with appropriate prompts for level A
 430 FOR I%=0 TO 3
 440 PROClight(I%,1,3)
 450 PRINT TAB(I%*7,31);P$(A,I%);
 460 NEXT I%
 470 PROClight(4,1,3)
 480 PRINT TAB(30,31);X; : PRINT TAB(34,31);",";Y;
 490 ENDPROC

 500 REM light
 510 DEF PROClight(KEY,BACK,TEXT)
 519 REM print key in BACK colour with label in TEXT colour
 520 COLOUR 128+BACK : COLOUR TEXT
 530 PRINT TAB(KEY*7,30);K$(KEY);
 540 COLOUR 128 : COLOUR 3
 549 REM make sure key is not still being held down
 550 REPEAT : UNTIL NOT INKEY(K(KEY))
 560 ENDPROC

 600 REM text
 610 DEF PROCtext(N)
 619 REM set text window of N lines up from bottom and clear buffers
 620 IF N<>0 THEN N=32-N
 630 VDU 28,0,31,39,N : *FX15,0
 639 REM if N=0 then window is whole of screen so don't clear it
 640 IF N<>0 THEN CLS
 650 ENDPROC

 700 REM initcursor
 710 DEF PROCinitcursor(XPOS,YPOS)
 719 REM set starting point for cursor
 720 X=XPOS : Y=YPOS : OX=-1 : OY=-1
 730 ENDPROC

 800 REM cursor
 810 DEFPROCcursor(M)
 819 REM M=0 means continue till f4 is pressed. M=1 is single step.
 820 GCOL3,3 : IF M=0 THEN S=1 ELSE S=4
 829 REM if cursor has moved then use PROCcross to change display
 830 IF OX<>X OR OY<>Y THEN PROCcross:OX=X:OY=Y
 840 IF INKEY(-122) AND X<1280-S THEN X=X+S
 850 IF INKEY(-26) AND X>=S THEN X=X-S
 860 IF INKEY(-42) AND Y>=65+S THEN Y=Y-S
 870 IF INKEY(-58) AND Y<1024-S THEN Y=Y+S
 879 REM if cursor is moving then add to step size and update the
         display of coordinates
 880 IF OX=X AND OY=Y THEN S=1 ELSE S=S+1 : PRINT TAB(30,31);"
";
     : PRINT TAB(30,31);X; : PRINTTAB(34,31);",";Y;
 889 REM if not single step mode then keep monitoring cursor keys
           unless f4 has been pressed while cursor is stationary
 890 IF M=0 AND ( S>1 OR NOT INKEY(-21) ) THEN 830
 899 REM in continuous mode cursor is removed when point is entered
 900 IF M=0 THEN OX=-1 : OY=-1 : PROCcross
 910 ENDPROC

1000 REM cross
1010 DEF PROCcross
1019 REM erase cross at OldX and OldY place new cross at X and Y
```

```
1020 MOVE 0,OY : DRAW 1280,OY : MOVE 0,Y : DRAW 1280,Y
1030 MOVE OX,0 : DRAW OX,1024 : MOVE X,0 : DRAW X,1024
1040 ENDPROC

1100 REM fill
1110 DEF PROCfill(X1,Y1,X2,Y2,X3,Y3)
1120 IF ((Y1 DIV 4)=(Y2 DIV 4) AND (Y1 DIV 4)=(Y3 DIV 4))
        THEN MOVEX1,Y1 : DRAWX2,Y2 : DRAWX3,Y3 : ENDPROC
1130 MOVE X1,Y1 : MOVE X2,Y2 : PLOT 85,X3,Y3
1140 ENDPROC

1200 REM save
1210 DEF PROCsave
1220 PROCtext(1)
1230 INPUT "FILENAME ",F$
1240 A$="*SAVE """+F$+""" 3000 7B00"
1249 REM use Command Line Interpreter to execute A$
1250 $32512=A$ : X%=0 : Y%=127 : CALL &FFF7
1260 PROCtext(0)
1270 ENDPROC

1300 REM load
1310 DEF PROCload
1320 PROCtext(1)
1330 INPUT "FILENAME ",F$
1340 A$="*LOAD """+F$+""" 3000"
1349 REM use Command Line Interpreter to execute A$
1350 $32512=A$ : X%=0 : Y%=127 : CALL &FFF7
1360 PROCtext(0)
1370 ENDPROC

1500 REM initdims
1510 DEF PROCinitdims
1520 ENDPROC

1600 REM initprompt
1610 DEF PROCinitprompt
1619 REM read prompts for keys for up to six levels of prompting
1620 DIM P$(5,3)
1630 FOR I%=0 TO 5 : FOR J%=0 TO 3
1640 READP$(I%,J%)
1650 NEXT J%:NEXT I%
1660 DATA"SAVE","LOAD","ERASE","ETC."
1670 DATA"","","",""
1680 DATA"","","",""
1690 DATA"","","",""
1700 DATA"","","",""
1710 DATA"","","",""
1720 ENDPROC

2000 REM key0
2010 DEF PROCkey0
2019 REM light key up when pressed & redisplay key
         as normal when finished
2020 PROClight(0,2,0)
2029 REM perform action approriate to level when pressed.
         If OL is different to LEVEL either by LEVEL change or
         resetting OL to -1 prompts are refreshed.
2030 IF LEVEL=0 THEN PROCsave : OL=-1
2040 IF LEVEL=1 THEN PROChisto : OL=-1
2050 PROClight(0,1,3)
2060 ENDPROC
```

```
2200 REM key1
2210 DEF PROCkey1
2219 REM see key0
2220 PROClight(1,2,0)
2230 IF LEVEL=0 THEN PROCload : OL=-1
2240 PROClight(1,1,3)
2250 ENDPROC

2400 REM key2
2410 DEF PROCkey2
2419 REM see key0
2420 PROClight(2,2,0)
2430 IF LEVEL=0 THEN PROCtext(1) : INPUT "ARE YOU SURE ",A$
     : PROCtext(0) : OL=-1 : IF A$="Y" OR A$="y" THEN CLG
2440 PROClight(2,1,3)
2450 ENDPROC

2600 REM key3
2610 DEF PROCkey3
2619 REM see key0
2620 PROClight(3,2,0)
2630 PROClight(3,1,3)
2640 ENDPROC

2800 REM key4
2810 DEF PROCkey4
2819 REM see key0 but same action taken at any LEVEL
2820 PROClight(4,2,0)
2830 PROCtext(1) : INPUT"NEW POSITION   X,Y ",X,Y : PROCtext(0)
2840 OL=-1 : PROClight(4,1,3)
2850 ENDPROC
```

Exercise 6.1
Change the ‘cursor’ procedure so that the standard cursor keys may be used with the shift key held down. In this case the crosswires are to move in character block steps about the graphics area.

The Four Diagram Construction Programs

Havng dealt with level 0 we now look at the four separate programs, each of which must be individually merged with listing 6.1 to produce the required DIAGRAM CONSTRUCTOR. Those who have system OS 1.0 and above will find that they do not have enough memory to hold and mn each of these four programs. What they must do is create the program, strip the REMs, save it, and reload it after typing PAGE = &1100. This will give them enough memory for loading and running. Now we shall look at each of the four programs in turn.

Histograms
Histograms (or bar-charts) can be constructed by our programs to any height and in any colour. Since we know how many addressable points are available on the screen we can formulate a method for calculating the spacing and

width of bars once we know their number. The first part of the 'histo' gram procedure (listing 6.2, which is used in conjunction with listing 6.1) is called by pressing f9 on level 1; f3 (ETC.) returns you to level 0. It immediately asks for the range of the vertical data (two integers in increasing order), creates the vertical scale, draws the horizontal and vertical axes, labels the vertical and then asks for the number of bars. Then the width of the bars and the size of the gaps between the neighbouring bars are calculated by a method similar to the scaling of the screen for two-dimensional graphics. For each bar the machine needs to know its height, and the inner and outer colours. On receiving these data the procedure uses the area-filling PLOT 85 option to colour in the bar. When the diagram is complete you must return to level 0 and SAVE the picture. Then load in the labelling program (listings 6.1 and 6.6), reLOAD the picture on level 0 and then on the lower levels add the necessary labels and headings.

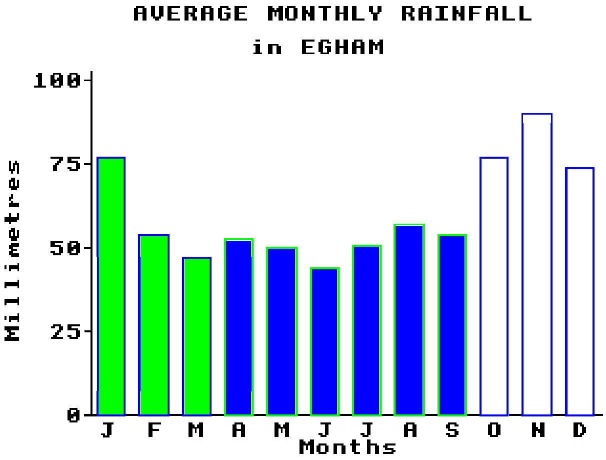

Figure 6.1

Example 6.1

Figure 6.1 for example, a diagram that presents the annual rainfall in Egham, was constructed (unlabelled) and saved, then reloaded with the general adjustment program (see later) and labels added until it was in the form above.

Listing 6.2

```
1500 REM initdims

1600 REM initprompt
1670 DATA"HISTOGRAM","","","ETC."

2000 REM key0
2040 IF LEVEL=1 THEN PROChisto : OL=-1
2050 PROClight(0,1,3)
2060 ENDPROC

2800 REM key4

3000 REM histo
3010 DEF PROChisto
3020 PROCtext(2) : INPUT"Range of vertical "YB," to "YT : PROCtext(0)
3030 IF YB>=YT THEN 3020
3040 YSCALE=640/(YT-YB)
3049 REM draw axes
3050 GCOL 0,3 : MOVE 208,864 : DRAW 208,208 : DRAW 1184,208
3060 YDIF=(YT-YB)/4 : TICK=YB
3069 REM put ticks & labels on y-axis
3070 FOR I%=1 TO 5 : TK=INT(TICK+0.5)
3080 Y=5*32*I%+48 : MOVE 208,Y : DRAW 196,Y : ROW=INT((1024-Y)/32)
3089 REM make sure label is sensible and correct length
3090 A$=STR$(TK) : IF LEN(A$)>3 THEN A$=LEFT$(A$,3)
     : IF TK>999 OR TK<-99 THEN A$="***"
3100 IF LEN(A$)<3 THEN REPEAT A$=" "+A$:UNTIL LEN(A$)=3
3110 PRINT TAB(3,ROW);A$ : TICK=TICK+YDIF
3120 NEXTI%
3130 PROCtext(2) : INPUT"NO. OF BARS "NB : PROCtext(0)
3139 REM calculate width of bars and gaps to fit x-axis
3140 XSCALE=976/NB : GAP=XSCALE/3 : WID=XSCALE-GAP
3149 REM get details and display each of the bars
3150 FOR I%=1 TO NB
3160 PROCtext(2) : PRINT"DATA FOR BAR ";I%;
3170 INPUT":"D,"INNER COL "C,"OUTER COL "OC : PROCtext(0)
3179 REM calculate bottom-left and top-right corners of block
3180 GCOL 0,C : X1=208+GAP/2+(I%-1)*XSCALE : X2=X1+WID : Y1=208
3190 IF D<=YB THEN Y2=212 : GOTO 3210
3200 D=D-YB : Y2=Y1+INT(D*YSCALE+0.5)
3209 REM fill in bar in inner colour
3210 PROCfill(X1,Y1,X2,Y2,X1,Y2) : PROCfill(X1,Y1,X2,Y2,X2,Y1)
3219 REM outline bar in outer colour
3220 GCOL 0,OC : MOVE X1,Y1 : DRAW X1,Y2 : DRAW X2,Y2
     : DRAW X2,Y1 : DRAW X1,Y1
3230 NEXT I%
3240 ENDPROC
```

Exercise 6.2
Write variations on this standard 'histo' procedure that can be substituted into the complete package as and when required. For example write a procedure that draws the histogram as a set of pairs of bars. The space between any two bars that form a pair should be half the distance between neighbouring bars that do not form a pair. Use this to construct diagrams that are similar to figure 6.2.

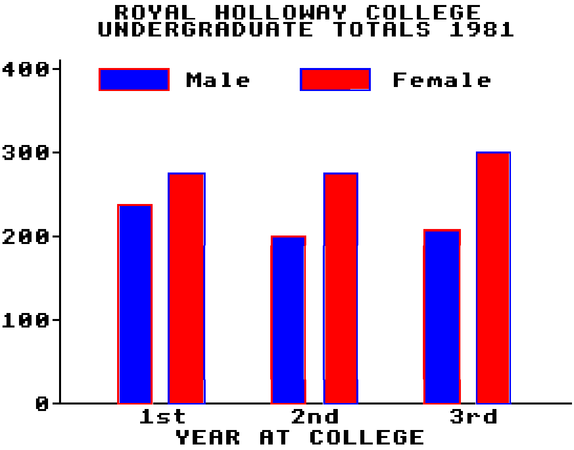

Figure 6.2

Example 6.2
In listing 6.3 we give an example of such a replacement 'histo' procedure. This version of 'histo' (using a variation on the fake-perspective cube procedure from chapter 1) produces an apparently three-dimensional graph. Two data values are requested for each bar, a MAXimum and a MINimum; the maximum bar is drawn behind the minimum bar. This program can be used to create charts similar to figure 6.3 which shows the monthly temperature variation in Egham.

Exercise 6.3
There are many, many more possible variations, for example drawing bars above and below a central line in order to display fluctuations in currency exchange rates. See the Money Programme on BBC2 for ideas. The fundamental notions we have introduced here should enable you to produce histograms to your own specifications.

Listing 6.3

```
1500 REM initdims

1600 REM initprompt
1670 DATA"HISTOGRAM","","","ETC."

2000 REM key0
2040 IF LEVEL=1 THEN PROChisto : OL=-1
2050 PROClight(0,1,3)
2060 ENDPROC

2200 REM key1

2400 REM key2

2600 REM key3
2630 LEVEL=(LEVEL+1) MOD 2
2640 PROClight(3,1,3)
2650 ENDPROC

2800 REM key4

3000 REM histo
3010 DEF PROChisto
3020 PROCtext(2) : INPUT"Range of vertical "YB," to "YT : PROCtext(0)
3030 IF YB>=YT THEN 3020
3040 YSCALE=640/(YT-YB)
3049 REM draw axes
3050 GCOL 0,3 : MOVE 208,864 : DRAW 208,208 : DRAW 1184,208
3060 YDIF=(YT-YB)/4 : TICK=YB
3069 REM put ticks & labels on y-axis
3070 FOR I%=1 TO 5 : TK=INT(TICK+0.5)
3080 Y=5*32*I%+48 : MOVE 196,Y : DRAW 208,Y : MOVE 248,Y+40
     : DRAW 168,Y-40 : ROW=INT((1024-Y)/32)
3089 REM"make sure label is sensible and correct length
3090 A$=STR$(TK) : IF LEN(A$)>3 THEN A$=LEFT$(A$,3)
     : IF TK>999 OR TK<-99 THEN A$="***"
3100 IF LEN(A$)<3 THEN REPEAT A$=" "+A$:UNTIL LEN(A$)=3
3110 PRINT TAB(3,ROW);A$ : TICK=TICK+YDIF
3120 NEXT I%
3130 PROCtext(2) : INPUT"NO. OF BARS "NB : PROCtext(0)
3139 REM calculate width of bars and gaps to fit x-axis
3140 XSCALE=976/NB : GAP=XSCALE/3 : WID=XSCALE-GAP
3149 REM get details and display each pair of bars
3150 FOR I%=1 TO NB
3160 PROCtext(2) : PRINT"DATA FOR BAR ";I%;
3170 INPUT":"D,D1 : PROCtext(0)
3180 C=1 : OC=3 : VDU19,2,4,0,0,0
3189 REM calculate bottom-left and top-right corners of blocks
3190 GCOL 0,C : X1=208+GAP+(I%-1)*XSCALE : X2=X1+WID : Y1=208
3200 IF D<=YB THEN Y2=212 : GOTO 3220
3210 D=D-YB : Y2=Y1+INT(D*YSCALE+0.5)
3219 REM draw back block in 3d then change colour and do front
         block over the top
3220 PROCfake3d(X1,Y1,X2,Y2) : C=2
3230 D1=D1-YB : Y2=Y1+INT(D1*YSCALE+0.5)
3240 X2=X2-WID/3 : X1=X1-WID/3 : Y1=Y1-WID/3 : Y2=Y2-WID/3
3250 PROCfake3d(X1,Y1,X2,Y2)
3260 NEXT I%
3270 ENDPROC
```

```
3400 REM fake3d
3410 DEF PROCfake3d(X1,Y1,X2,Y2)
3419 REM draw rectangle defined by coordinates of its diagonal
3420 PROCquad(X1,Y1,X1,Y2,X2,Y2,X2,Y1)
3429 REM add rhombus to top and side to simulate a 3d box
3430 PROCquad(X2,Y1,X2,Y2,X2+WID/3,Y2+WID/3,X2+WID/3,Y1+WID/3)
3440 PROCquad(X1,Y2,X2,Y2,X2+WID/3,Y2+WID/3,X1+WID/3,Y2+WID/3)
3450 ENDPROC

3500 REM quad
3510 DEF PROCquad(XA,YA,XB,YB,XC,YC,XD,YD)
3519 REM fill in a quadrilateral in colour C
          and outline it in colour OC
3520 GCOL 0,C : MOVE XA,YA : MOVE XB,YB
3530 PLOT 85,XD,YD : PLOT 85,XC,YC
3540 GCOL 0,OC : DRAW XD,YD : DRAW XA,YA
3550 DRAW XB,YB : DRAW XC,YC
3560 ENDPROC
```

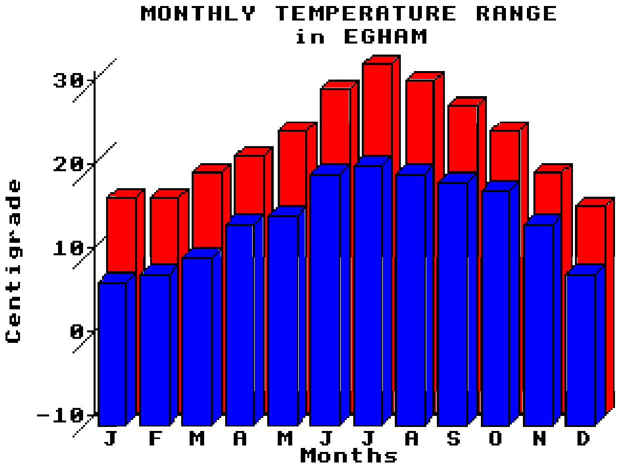

Figure 6.3

Pie-charts

The pie-chart is a favourite with economists and biologists who delight in telling us how big each slice of our capital expenditure cake is, or alternatively which fungi are growing on it. The usual requirements of a pie-chart program are that it should draw ‘pies’ of variable radii, it must be possible for some slices to be pulled out from the centre, and provision must be made for these slices to be filled-in or cross-hatched. A pie-chart and associated procedures are given in listing 6.4. It is entered by pressing f0 on level 1; pressing f3 (ETC.) returns you to level 0. The program first requires the number of pie-slices and the individual data values; the sum of values is used to establish an angular scale for the pie-chart. The 'pie' is centred by the crosswires, that is, by using the cursor keys to position it, and the coordinates are entered with f4 (showing white). The radius ofthe pie-chart (in addressable points) is then INPUT. Each slice is centred with the cursor; any displacement ofthe cursor from the centre ofthe 'pie' is treated as a distance along the bisector of the slice and not as an absolute position. With each new section the cursor re-appears at the original centre ofthe 'pie'. Then the program enquires if you wish to hatch the 'pie' (x, y, b, n?) in the *x*-direction, *y*-direction, both or neither (this is explained in a moment); should you wish to hatch a slice then the program asks for further information about the position of the hatching lines and the distance between them. It then requests the inner and outer colour of the pie-slice and finally draws it. Figure 6.4 was generated using this procedure, the 'hatch' ing procedure below and the labelling program. After the picture is complete you return to level 1 .

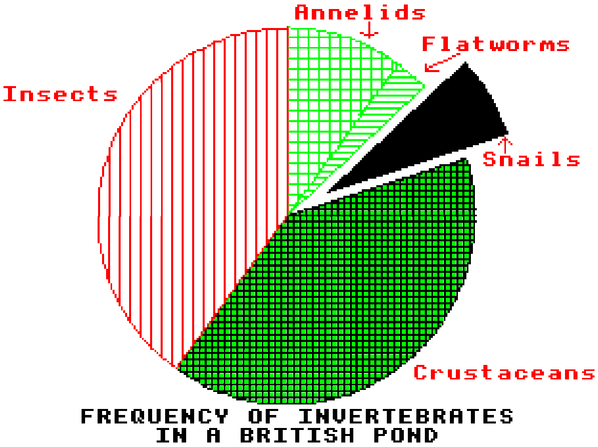

Figure 6.4

Hatching

Hatching latching the area of a pie-slice involves the intersection of a line with the boundaries of the slice. To make the calculations simpler we shall hatch only with lines in the horizontal or vertical directions, or both. Furthermore we only hatch 'pies' that subtend angles less than or equal to π radians (180 degrees)

Listing 6.4

```
1500 REM initdims
1520 DIM D(20),Z(4)
1530 ENDPROC

1600 REM initprompt
1670 DATA"PIE-CHART","","","ETC."

2000 REM key0
2040 IF LEVEL=1 THEN PROCpie : OL=-1
2050 PROClight(0,1,3)
2060 ENDPROC

2200 REM key1

2400 REM key2

2600 REM key3
2630 LEVEL=(LEVEL+1) MOD 2
2640 PROClight(3,1,3)
2650 ENDPROC

2800 REM key4

3000 REM pie-chart
3010 DEF PROCpie
3020 PROCtext(2) : INPUT"No. OF SEGMENTS "NB : SUM=0
3029 REM get data for all sections
3030 FOR I%=1 TO NB
3040 PRINT"DATA ";I%; : INPUT": "D(I%) : SUM=SUM+D(I%)
3050 NEXT I%
3060 PROCtext(0)
3069 REM use the cursor to indicate the centre of the pie-chart
3070 PROCprompt(5) : PRINT TAB(0,31);"CENTRE PIE";
3080 PROClight(4,3,1) : PROCcursor(0) : PROClight(4,1,3)
3090 XC=X : YC=Y
3100 PROCtext(1) : INPUT"RADIUS (in addressable points) "R :
PROCtext(0)
3110 SCALE=2*PI/SUM : A1=PI/2
3119 REM if cursor is moved from centre movement is treated as
         along bisector of angle subtended by segment
3120 FOR I%=1 TO NB
3130 PROCtext(1) : PRINT"CENTRE SEGMENT ";I%; : PROCtext(0)
3140 PRINT TAB(30,31);X; : PRINT TAB(34,31);",";Y;
3150 PROCinitcursor(XC,YC) : PROClight(4,3,1) : PROCcursor(0) :
PROClight(4,1,3)
3159 REM make sure pie joins up and get scale size of segment
3160 IF I%=NB THEN A2=-3*PI/2 : ANG=A1-A2
        ELSE ANG=SCALE*D(I%) : A2=A1-ANG
3170 IF X=XC AND Y=YC THEN 3200
3179 REM deal with displacement
3180 A3=A1-ANG/2 : DIST=SQR((X-XC)^2+(Y-YC)^2)
3190 X=INT(XC+DIST*COS(A3)+0.5) : Y=INT(YC+DIST*SIN(A3)+0.5)
3199 REM is hatching to be along the 'x' axis, 'y' axis, 'b'oth
                or 'n'ot at all
3200 PROCtext(1) : INPUT"HATCH (x,y,b,n) "H$
     : IF ASC(H$)<96 THEN H$=CHR$(ASC(H$)+32)
3209 REM what spacing between lines and what initial offset
3210 IF H$<>"n" THEN INPUT "JUMP"JUMP,"FROM "FROM
3220 INPUT"INNER COLOUR",IC,"OUTER COLOUR",OC
```

```
3229 REM fill in segment
3230 GCOL 0,IC : X1=X+INT(R*COS(A1)+0.5) :
Y1=Y+INT(R*SIN(A1)+0.5)
3240 MOVE X1,Y1 : ADIF=-10/R
3250 FOR T=A1+ADIF TO A2 STEP ADIF
3260 MOVE X,Y : PLOT 81,COS(T)*R,SIN(T)*R
3270 NEXT T
3279 REM make sure filled in to end of segment
3280 MOVE X,Y : XS=INT(COS(A2)*R+0.5) : YS=INT(SIN(A2)*R+0.5)
     : PLOT81,XS,YS
3289 REM draw outline round segment
3290 GCOL 0,OC : MOVE X,Y : DRAW X1,Y1
3300 FOR T=A1+ADIF TO A2 STEP ADIF
3310 DRAW X+COS(T)*R,Y+SIN(T)*R
3320 NEXT T
3329 REM make sure outline goes to edge of segment
3330 DRAW X+XS,Y+YS : DRAW X,Y
3340 IF H$="n" THEN 3370 ELSE A2S=A2
3349 REM deal with hatching if the segement is more than half
circle
         treat as two parts
3350 IF ANG>PI THEN A2=A1-PI : X2=2*X-X1 : Y2=2*Y-Y1 :
PROChatch(H$) : X1=X2 : Y1=Y2 : A1=A2 : A2=A2S
3360 PROCtext(0) : X2=X+XS : Y2=Y+YS : PROChatch(H$)
3370 A1=A2
3380 NEXT I%
3390 ENDPROC

3400 REM hatch
3410 DEF PROChatch(H$)
3419 REM easy way to get both ways
3420 IF H$="b" THEN PROChatch("x") : PROChatch("y") : ENDPROC
3430 IF H$="y" THEN PZ=X : PT=Y : Z1=X1 : T1=Y1 : Z2=X2 : T2=Y2
3440 IF H$="x" THEN PZ=Y : PT=X : Z1=Y1 : T1=X1 : Z2=Y2 : T2=X2
3449 REM find the max. and min. coordinates for lines which pass
         through segment.
3450 T=PI/2 : MAX=0 : MIN=0
3460 IF H$="x" THEN V=COS(A1) ELSE V=SIN(A1)
3470 IF MAX < V THEN MAX=V ELSE IF MIN > V THEN MIN=V
3480 IF T>A1 THEN REPEAT : T=T-PI/2 : UNTIL T<=A1
3490 IF T<A2 THEN 3530
3500 IF H$="x" THEN V=COS(T) ELSE V=SIN(T)
3510 IF MAX < V THEN MAX=V ELSE IF MIN > V THEN MIN=V
3520 T=T-PI/2 : GOTO3490
3530 IF H$="x" THEN V=COS(A2) ELSE V=SIN(A2)
3540 IF MAX < V THEN MAX=V ELSE IF MIN > V THEN MIN=V
3550 NMIN=INT(INT(R*MIN+1)/JUMP)*JUMP+FROM
3559 REM for lines which cross segment find intersections
         with radii and arc
3560 FOR E=NMIN TO MAX*R STEP JUMP
3570 C=0 : DENOM=T1-PT : IF DENOM=0 THEN 3600
3580 MU=E/DENOM : IF MU<0 OR MU>1 THEN 3600
3590 C=C+1 : Z(C)=PZ+MU*(Z1-PZ)
3600 DENOM=T2-PT : IF DENOM=0 THEN 3640
3610 MU=E/DENOM : IF MU<0 OR MU>1 THEN 3640
3620 C=C+1:Z(C)=PZ+MU*(Z2-PZ)
3629 REM if more than two points of intersection found,
         delete duplicates
3630 IF C=2 AND Z(1)=Z(2) THEN C=1
3640 IF C<>2 THEN 3670
3650 IF H$="y" THEN MOVE Z(1),E+PT : DRAW Z(2),E+PT : GOTO 3730
3660 IF H$="x" THEN MOVE E+PT,Z(1) : DRAW E+PT,Z(2) : GOTO 3730
```

```
3670 DISC=R*R-E*E : IF DISC<0 THEN 3730
3680 DISC=INT(SQR(DISC)+0.5)
3690 ZZ=PZ+DISC : AZ=DISC : PROCin : IF IN THEN C=C+1 : Z(C)=ZZ
3700 ZZ=PZ-DISC : AZ=-DISC : PROCin : IF IN THEN C=C+1 : Z(C)=ZZ
3710 IF C>2 AND Z(1)=Z(2) THEN Z(2)=Z(3)
3720 IF C>=2 THEN 3650
3730 NEXT E
3740 ENDPROC

3800 REM in
3810 DEF PROCin
3819 REM if angle lies between angles of ends of segment then
         point of intersection is on the arc of the segment
3820 IF H$="x" THEN BZ=E : EZ=AZ ELSE BZ=AZ : EZ=E
3829 REM find angle from centre to point of intersection
3830 IF BZ=0 THEN PHI=-PI/2 : IF EZ>0 THEN PHI=-PHI
3840 IF BZ<>0 THEN PHI=ATN(EZ/BZ) : IF BZ<0 THEN PHI=PHI-PI
3849 REM IN is true if PHI is between angles of edges
3850 IN=(PHI<=A1) AND (PHI>=A2)
3860 ENDPROC
```

at the centre. For obtuse angles the ‘pie’ is treated as two pieces, the first subtending π radians at the centre. The ‘pie’ procedure enquires whether the hatching is to be horizontal (answer ‘x'), vertical (answer ‘y'), both ways (answer ‘b') or neither (answer ‘n').

The pie sections we are considering are each bounded by two line segments and a circular arc. We must find which part of a hatching line (if any) lies inside this segment. Because the ‘pie’ does not subtend an angle greater than p radians at its centre there are only four possibilities:

(1) A line may miss the pie altogether.
(2) It may intersect the arc at two points.
(3) It may intersect the arc and one ofthe line segments.
(4) It may intersect both line segments.

The special cases where the line coincidently cuts the arc and a line segment at the same point may be included in one of the above four possibilities. The explanation of the hatching algorithm is given with reference to horizontal hatching; the vertical follows in an equivalent manner. We first find the MAXimum and MINimum y-values ofpoints within the ‘pie’ section. Then we consider all horizontal hatching lines with equations of the form Y = k * JUMP + FROM between these limits (0 ≤ FROM ≤ JUMP – 1). For each hatching line we calculate the two points of intersection with the extended line segments and then check whether their MU values lie between 0 and 1, that is, whether the intersection is between the centre of the circle and the arc. Next we find the two points of intersection of the hatching line with the complete circle that contains the arc and then check whether they lie on the arc. From these we can find the two points of intersection of the pie section and the hatching line, and these are then joined. This whole process is programmed in listing 6.4 and an example of its use is given in figure 6.4 above. Note that if we set the JUMP to a number that is not the size of a whole number of pixels (in addressable points) we get unusual candy-stripe or dotted hatching.

Graphs

As our third exainple of graphical data presentation we must consider scientific graphs of functions and graphs of discrete points, called by f9 at level 1 (listings 6.1 and 6.5). Such diagrams require coordinate axes that need be neither of fixed size nor offixed scale. The program requests the lower and upper *x*-value and *y*-value of the data. Then a standard method is used to decide on the placing of a particular axis: if zero should lie in the range of the graph then the axis passes through that point, otherwise it lies on the edge of the graphics area, closest to zero. Five marks are then placed along each axis and if automatic labelling is specified then the corresponding scale value is written close to each mark. The need for accuracy in scientific graphs necessitates the use of as many characters as possible across the screen. The BBC Model B has only 40 characters across the screen and 32 up it, so previously loaded ‘thin’ characters (character set 4 from chapter 5) are placed two per character block, enabling us to draw 80 characters on each line. When numbers are to be printed as ‘thin’ ‘label’s they need to be converted into strings and made consistent in lgth and/or decimal accuracy. This is achieved by the procedure ‘number’ (listing 6.5).

Exercise 6.4

Write an extended ‘number’ procedure that allows you to specify the format of the string to be printed. One way of doing this is to enter a string that contains a template for the number format, for example the string ‘##.###’ could specify a number with two digits before the decimal point and three decimal places after it. Also see the @% option in the user manual.

The choice is now offered between entering a functional representation of points on a continuous curve, and entering a set of discrete data points to be joined in a saw-tooth twe pattern by straight lines. In the functional section of the procedure the program asks for an algebraic expression for the function, which may include the standard in-built functions (like SIN or COS) as well as Nur own functions. The height of the point on the curve above each pixel point on the X-axis is calculated, and these points are joined by lines.

In the discrete section the number of data points is INPUT, followed by the individual X and Y coordinates which are sorted into ascending order of the X coordinate. Consecutive points are then joined by lines. One example of each type of diagram is given. Figure 6.5 shows a typical continuous cosine curve and figure 6.6 shows discrete scientific data that illustrate the pH levels of a river.

Listing 6.5

```
1500 REM initdims
1520 DIM X(100),Y(100)
1530 ENDPROC

1600 REM initprompt
1670 DATA"GRAPH","","","ETC."

2010 DEF PROCkey0
2040 IF LEVEL=1 THEN PROCgraph : OL=-1
2050 PROClight(0,1,3)
2060 ENDPROC

2200 REM key1

2400 REM key2

2600 REM key3
2630 LEVEL=(LEVEL+1) MOD 2
2640 PROClight(3,1,3)
2650 ENDPROC

2800 REM key4

3000 REM graph
3010 DEF PROCgraph
3020 PROCtext(2)
3030 REPEAT : INPUT"X GOES FROM "XB,"TO "XT : UNTIL XT>XB
3040 REPEAT : INPUT"Y GOES FROM "YB,"TO "YT : UNTIL YT>YB
3050 INPUT"AUTOMATIC LABELLING (Y/N) ",L$ : PROCtext(0)
3060 XSCALE=1024/(XT-XB) : YSCALE=704/(YT-YB)
3069 REM draw axes through origin or on side closest to origin
3070 IF YT<0 THEN YO=896 ELSE IF YB>0 THEN YO=192
        ELSE YO=INT(-YB*YSCALE+192.5)
3080 IF XT<0 THEN XO=1152 ELSE IF XB>0 THEN XO=128
        ELSE XO=INT(-XB*XSCALE+128.5)
3090 MOVE XO,192 : DRAW XO,896 : MOVE 128,YO : DRAW 1152,YO
3100 XDIF=(XT-XB)/4 : YDIF=(YT-YB)/4
3109 REM put five ticks along each axis with 'thin' labels
3110 X=XB : Y=YB : FOR J=1 TO 5
3120 PX=INT((X-XB)*XSCALE+128.5) : PY=YO
3130 MOVE PX,PY-8 : DRAW PX,PY+8
3139 REM calculate text positions
3140 TX=PX DIV 32 -1 : TY=(1024-PY) DIV 32+1
3150 IF L$="Y" THEN PROCthin(1,TX,TY,STR$(X))
3160 PX=XO : PY=INT((Y-YB)*YSCALE+192.5)
3170 MOVE PX-8,PY : DRAW PX+8,PY
3180 TX=PX DIV 32 +1 : TY=(1024-PY) DIV 32-1
3190 IF L$="Y" THEN PROCthin(1,TX,TY,STR$(Y))
3200 X=X+XDIF : Y=Y+YDIF : NEXT J
3210 PROCtext(2)
3220 REPEAT : INPUT"CONTINUOUS OR DISCRETE ",D$ : UNTIL D$="C" OR D$="D"
3230 IF D$="D" THEN 3330
3239 REM section to plot graph of a function
3240 INPUT"F(X): Y="F$
3249 REM evaluate function for X to find point on curve
3250 X=XB : Y=EVAL(F$) : IY=INT((Y-YB)*YSCALE+192.5)
3260 MOVE 128,IY
3269 REM repeat for values of X one pixel apart
3270 FOR I%=128 TO 1152 STEP 4
```

```
3280 X=(I%-128)/XSCALE+XB
3290 Y=EVAL(F$) : IY=INT((Y-YB)*YSCALE+192.5)
3300 DRAW I%,IY
3310 NEXT I% : X=640 : Y=512
3320 ENDPROC
3329 REM come here if points to be joined are to be input
3330 INPUT"NO. OF POINTS "NP
3339 REM get all points
3340 FOR I%=1 TO NP
3350 PRINT"X(";I%;"),Y(";I%;") "; : INPUTX(I%),Y(I%)
3360 NEXT I%
3369 REM use simple bubble-sort to get points in ascending order
         of X-coordinate
3370 FOR I%=1 TO NP-1 : FOR J%=I%+1 TO NP
3380 IF X(J%)<X(I%) THEN T=X(J%) : X(J%)=X(I%) : X(I%)=T
     : T=Y(J%) : Y(J%)=Y(I%) : Y(I%)=T
3390 NEXT J% : NEXT I%
3399 REM find scale coordinates of first point and draw symbol
3400 X=INT((X(1)-XB)*XSCALE+128.5) : Y=INT((Y(1)-
YB)*YSCALE+192.5)
3410 MOVE X,Y : PROCsymbol(X,Y)
3419 REM repeat for other points joining them up
3420 FOR I%=2 TO NP
3430 X=INT((X(I%)-XB)*XSCALE+128.5) : Y=INT((Y(I%)-
YB)*YSCALE+192.5)
3440 DRAW X ,Y : PROCsymbol(X,Y)
3450 NEXT I%
3460 ENDPROC

3500 REM symbol
3510 DEF PROCsymbol(X,Y)
3519 REM draw a little square round X,Y to mark it
3520 MOVE X+8,Y+8 : DRAW X-8,Y+8 : DRAW X-8,Y-8
     : DRAW X+8,Y-8 : DRAW X+8,Y+8 : MOVE X,Y
3530 ENDPROC

4000 REM thin
4010 DEF PROCthin(M,X,Y,A$)
4019 REM output a numeric string using thin characters
4020 LOCAL I%,A%,P$
4030 A%=2^(4+M) : VDU 5
4040 MOVE X*A%,1023-Y*32
4050 FOR I%=1 TO LEN(A$) : P$=MID$(A$,I%,1)
4060 IF P$>="0" AND P$<="9" THEN VDU (80+ASC(P$)) : GOTO 4080
4070 IF P$="+" THEN VDU 144 ELSE IF P$="-" THEN VDU 145
     ELSE VDU 146
4080 MOVE (X+I%/2)*A%,1023-Y*32
4090 NEXT I%
4100 VDU 4
4110 ENDPROC
```

Exercise 6.5

It has been noted that the only requirement fur such graphs is a set of coordinates in ascending order of X which are then joined up. This set can be created in any manner: by a series of READ statements or by a multi-line calculation in a user-defined function FNf, which can be drawn by simply typing Y = f(X) instead of just using functions provided by the system. DEFine an FN that allows the graph of SIN(X)/X to be drawn; avoid the calculation of SIN (0)/0!

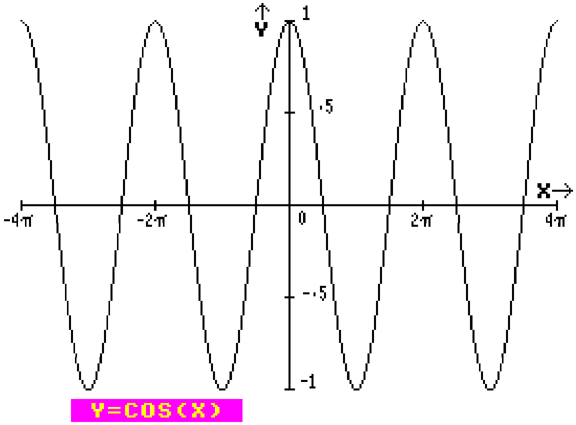

Figure 6.5

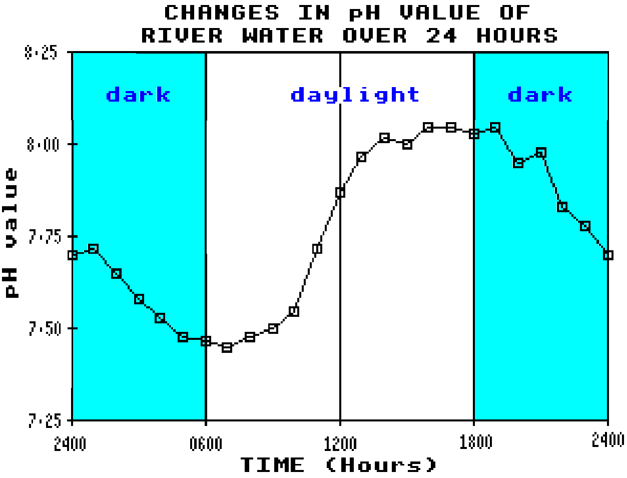

Figure 6.6

Diagram adjustment and labelling

Having drawn diagrams we now need simple control over the superimposition of labels and other graphics objects on them. This requires procedures for drawing lines and shapes, even perhaps for filling in the shapes (listing 6.6). On level 1 we have the options SKETCH (f9), LINE (fl), SHAPE (f2), ETC. (f3) and the Cursor adjustment (f4). SKETCH enables you to sketch in small details. The cursor keys move the crosswires around the screen, and if ‘P’ is pressed simultaneously then a trail is left behind (compare with listing 1.8). fD returns you to level 1. It would be very tedious to sketch in every pixel for a large block of the screen. Instead we use a set of procedures that draw lines and draw and/or fill triangles, boxes, polygons and circles. The LINE option uses the ‘line’ procedure which specifies two points on the screen by using the cursor, and then draws a line between them. It also demonstrates some of the other procedures that are provided for this part of the diagram package, such as ‘mark’ which places a small cross on the screen to show previous positions of the cursor.

SHAPE takes us down to level 3 with options DRAW (f9), FILL (fl) or BOTH (f3) which specifies the mode of colouring: outline only (f0), solid area only (fl) or both (f2). This leads on directly to level 4 with options TRIANG (f0), BOX (fl), CIRCLE (f2) and POLY (f3). The first two options allow us to draw a triangle or a quadrilateral with corners entered via the cursor. CIRCLE draws a circle with centre and radius given by the cursor. POLY draws a regular polygon with centre and radius specified by the cursor and the angle that one of the vertices makes with the horizontal given as a multiple of PI, and then returns to level 1. ETC. (f3) leads directly to level 2 which defines labels and the colours in use. LABEL (f0) enters the ‘label’ procedure which asks for Normal, Thin or Graphics characters and the string to be printed. Note that references are regularly made to the level 0 procedures, and to the ‘thin’ procedure (taken liom listing 5.4). Naturally we must first place the ‘thin’ characters in character set 4 position between &C00 and &D00. Obviously you can create other characters (such as a ‘thin’ π) and as long as you place them in set 4 they can be printed out by ‘label' .

COLOUR (fl) changes the foreground and/or background colours of labels. GCOL (f2) changes the Graphics and outline colours. ETC. (f3) leads you back to level 0. In most of these levels if the pseudo-soft key f4 is coloured red then you can INPUT a new position for the cursor rather than use the cursor keys.

Exercise 6.6

Draw a picture ofyour BBC micro, or perhaps a scene like that given at theend of chapter 1. Adapt your program from exercise 1.3 for use as a procedure to draw an *n*-sided polygon by using the ‘cursor’ to enter the n points.

If you have a graphics pad then you can copy rough sketches from the pad into the machine. You should then write programs to tidy up these pictures, that is to straighten lines and to smooth out curves.

Listing 6.6

```
1500 REM initdims

1600 REM initprompt

1670 DATA"SKETCH","LINE","SHAPE","ETC."
1680 DATA"LABEL","COLOUR","GCOL","ETC."
1690 DATA"DRAW","FILL","BOTH",""
1700 DATA"TRIANG","BOX","CIRCLE","POLY"

2000 REM key0
2040 IF LEVEL=1 THEN PROCsketch
2050 IF LEVEL=2 THEN PROClabel
2060 IF LEVEL=3 THEN OUTLINE=TRUE : FILL=FALSE : LEVEL=4 : GOTO 2080
2070 IF LEVEL=4 THEN PROCtriang : LEVEL =1
2080 PROClight(0,1,3)
2090 ENDPROC

2200 REM key1
2240 IF LEVEL=1 THEN PROCline : OL=-1
2250 IF LEVEL=2 THEN PROCtext(1) : INPUT "BACKGROUND ",BC,
     "TEXT COLOUR ",TC : PROCtext(0) : OL=-1
2260 IF LEVEL=3 THEN OUTLINE=FALSE : FILL=TRUE: LEVEL=4 : GOTO 2280
2270 IF LEVEL=4 THEN PROCbox : LEVEL=1
2280 PROClight(1,1,3)
2290 ENDPROC

2400 REM key2
2440 IF LEVEL=1 THEN LEVEL=3 : GOTO 2480
2450 IF LEVEL=2 THEN PROCtext(1) : INPUT "GCOL ",GC,
     "OUTLINE COLOUR ",OC : PROCtext(0) : OL=-1
2460 IF LEVEL=3 THEN OUTLINE=TRUE : FILL=TRUE : LEVEL=4 : GOTO 2480
2470 IF LEVEL=4 THEN PROCcircle : LEVEL=1
2490 ENDPROC

2600 REM key3
2630 IF LEVEL<3 THEN LEVEL=(LEVEL+1) MOD 3
2640 IF LEVEL=4 THEN PROCtext(1) : INPUT "No. OF SIDES",N,"ANGLE PI*"A
      : ANG=PI*A : PROCtext(0) : PROCpoly(N,ANG) :  LEVEL=1
2650 PROClight(3,1,3)
2660 ENDPROC

2800 REM key4

3000 REM sketch
3010 DEF PROCsketch
3020 REPEAT
3030 PROCcursor(1)
3040 IF INKEY(-56) THEN GCOL 3,GC : PLOT69,X,Y : GCOL 3,3
3050 PROCcross : OX=-1 : OY=-1 : PROCcross
3060 UNTIL INKEY(K(0))
3070 ENDPROC

3100 REM label
3110 DEF PROClabel
3120 PROClight(4,3,1) : PROCcursor(0) : PROClight(4,1,3)
3130 PROCtext(1) : INPUT"LABEL TYPE (N,T,G) ",T$
     : INPUT"LABEL ",A$ : PROCtext(0)
3140 COLOUR 128+BC : COLOUR TC : GCOL 0,TC
3150 IF T$="T" THEN PROCthin(1,X/32,(1024-Y)/32,A$) : GOTO 3180
3160 IF T$="N" THEN PRINT TAB(X/32,(1024-Y)/32);A$
3170 IF T$="G" THEN MOVE X,Y : VDU 5 : PRINT A$ : VDU 4
```

```
3180 OL=-1
3190 ENDPROC

3200 REM point
3210 DEF PROCpoint(A$)
3220 PROClight(4,3,1) : PROCtext(1) : PRINTA$; : PROCtext(0)
3230 PROCcursor(0) : SOUND1,-15,200,1
3240 PROClight(4,1,3)
3250 ENDPROC

3300 REM mark
3310 DEF PROCmark(X,Y)
3320 GCOL3,3 : MOVE X-12,Y-12 : DRAW X+12,Y+12
3330 MOVE X+12,Y-12 : DRAW X-12,Y+12
3340 ENDPROC

3400 REM line
3410 DEF PROCline
3420 PROCpoint("START") : A=X : B=Y : PROCmark(A,B)
3430 PROCpoint("END") : PROCmark(A,B)
3440 MOVE A,B : GCOL 0,GC : DRAW X,Y
3450 ENDPROC

3500 REM triang
3510 DEF PROCtriang
3520 PROCpoint("FIRST") : A=X : B=Y : PROCmark(A,B)
3530 PROCpoint("SECOND") : C=X : D=Y : PROCmark(C,D)
3540 PROCpoint("FINAL") : PROCmark(A,B) : PROCmark(C,D)
3550 GCOL0,GC : IF FILL THEN PROCfill(X,Y,A,B,C,D) : GCOL 0,OC
3560 IF OUTLINE THEN MOVE X,Y : DRAW A,B : DRAW C,D : DRAW X,Y
3570 ENDPROC

3600 REM box
3610 DEF PROCbox
3620 PROCpoint("FIRST") : A=X : B=Y : PROCmark(A,B)
3630 PROCpoint("SECOND") : C=X : D=Y : PROCmark(C,D)
3640 PROCpoint("THIRD") : E=X : F=Y : PROCmark(E,F)
3650 PROCpoint("FINAL") : PROCmark(A,B) : PROCmark(C,D)
     : PROCmark(E,F)
3660 GCOL0,GC : IF FILL THEN PROCfill(X,Y,A,B,E,F)
     : PROCfill(A,B,E,F,C,D) : GCOL 0,OC
3670 IF OUTLINE THEN MOVE X,Y : DRAW A,B : DRAW C,D : DRAW E,F : DRAW X,Y
3680 ENDPROC

3700 REM poly
3710 DEF PROCpoly(N,PHI)
3720 PROCpoint("CENTRE") : A=X : B=Y : PROCmark(A,B)
3730 PROCpoint("RADIUS") : PROCmark(A,B)
3740 R=SQR((X-A)^2+(Y-B)^2)
3750 OX=A+COS(PHI)*R : OY=B+SIN(PHI)*R
3760 FOR I=PHI TO 2*PI+PHI+PI/N STEP 2*PI/N
3770 X=A+COS(I)*R : Y=B+SIN(I)*R
3780 IF FILL THEN GCOL 0,GC : PROCfill(A,B,X,Y,OX,OY)
3790 IF OUTLINE THEN GCOL 0,OC : MOVE OX,OY : DRAW X,Y
3800 OX=X : OY=Y
3810 NEXT I : PROCinitcursor(A,B)
3820 ENDPROC

3900 REM circle
3910 DEFPROCcircle
3920 PROCpoly(100,0)
3930 ENDPROC
```

```
4000 REM thin
4010 DEF PROCthin(M,X,Y,A$)
4020 LOCAL I%,A%,P$
4030 A%=2^(4+M) : VDU 5
4040 MOVE X*A%,1023-Y*32
4050 FORI%=1 TO LEN(A$) : P$=MID$(A$,I%,1)
4060 IF P$>="0" AND P$<="9" THEN VDU (80+ASC(P$)) : GOTO 4080
4070 IF P$="+" THEN VDU 144 ELSE IF P$="-" THEN VDU 145 ELSE VDU
146
4080 MOVE (X+I%/2)*A%,1023-Y*32
4090 NEXT I%
4100 VDU 4
4110 ENDPROC
```

Complete Programs

To clarify the use of the programs given in this chapter we have underlined typical responses in the examples below. All the listings should be loaded with PAGE = &1100. Also use the REM stripper.

I Listings 6.1 and 6.2.
Type f3 (ETC.), f0 (histogram)
Rang of vertica1 0 to 100, number of bars 6
Data for bar 1: 75, inner colour 1, outer colour 3
Data for bar 2: 60, inner colour 0, outer colour 2
Data for bar 3: 88, inner colour 2, outer colour 1
Data for bar 4: 23, inner colour 3, outer colour 1
Data for bar 5: 17, inner colour 3, outer colour 1
Data for bar 6: 97, inner colour 1, outer colour 2
f3 (ETC.), f0 (SAVE), Filename? PIC1

II Listings 6.1 and 6.3.

Type f3 (ETC.), f0 (histogram/second type)
Range of vertical 0 to 200, number of bars 4
Data for bar 1: 166, 84
Data for bar 2: 100, 44
Data for bar 3: 80, 33
Data for bar 4: 40, 10
f3 (ETC.), f0 (SAVE), Filename? PIC2

III Listings 6.1 and 6.4.

Type f3 (ETC.), f0 (pie-chart), number of segments 4
Data 1: 4

Data 2: 3
Data 3: 2
Data 4: 3
Centre pie by using cursors, and enter with f4
Radius in addressable points 400
Centre segment 1 by using cursors and f4
Hatch? X, JUMP 8 FROM 1, inner colour 3, outer colour 1
Centre segment 2 using curors and f4
Hatch? N, , inner colour 0, outer colour 3
Centre segment 3 by using cursors and f4
Hatch? B, JUMP 12 FROM 0 ,inner colour 1, outer colour 3
Centre segment 4 by using cursors and f4
Hatch? Y, JUMP 11 FROM 0, inner colour 0, outer colour 2
f3 (ETC.), f0 (SAVE), Filename? PIC3

IV Listings 6.1 and 6.5

Type f3 (ETC.), f0 (graph)
X goes from –10 to 10
Y goes from –1 to 1
Automatic labelling (Y/N) N
Continuous or Discrete, C
F(X): Y = COS(X)
f3 (ETC.), f0 (SAVE), Filename? PIC4

V Listing 6.1 and 6.6

Type f1 (LDAD), Filename? PIC1
f3 (ETC.), f3 (ETC.), note ETC. twice!
f1 (COLOUR), background 3, text colour 1, f3, f3
f2 (SHAPE), f2 (BOTH), f0 (TRIANG)
FIRST (use cursor and f4)
SECOND (use cursor and f4)
FINAL (use cursor and f4), f3
f0 (LABEL), cursor to position and f4
label type (N, T, G)? N
LABEL: HELLO FOLKS
f3 (ETC.) f0 (SAVE), Filename? PIC5