whereas mode 5 needs (160×256)/4=10 Kbytes of memory. By choosing mode 5 we allow ourselves an extra 10 Kbytes of memory for our program. An added bonus is that mode 2 is not available on model A machines, but mode 5 is!

The construction of user-defined characters is simple on the BBC. Each character is made up of eight numbers representing the decimal equivalents of each row. The character designs for the mines, detector, and assistant are given.

The command VDU23 allows the programmer to define characters with ASCII codes from 226 to 255. The second number in the VDU command denotes which code you wish to assign to the shape defined by the last eight numbers.

VDU23,224,0,0,56,254,254,124,0,0

defines CHR$(224) as the mine shape. To print this character we simply issue the instruction PRINT CHR$(224). The following procedure defines the three characters to be used in the game.

```
2380 DEF PROCdefine_characters
2390 REM ** MINES **
2400 VDU23,224,0,0,56,254,254,124,0,0
2410 REM ** MINE DETECTOR **
2420 VDU23,225,231,195,189,36,36,189,195,231
2430 REM ** ASSISTANT **
2440 VDU23,226,56,56,16,124,186,170,40,108
2450 ENDPROC
```

SCREEN LAY-OUT

Once the shapes have been defined we can print them onto the screen. This is most easily done using the PRINT TAB(X,Y) command. In mode 5 there are 32 rows, each with 20 characters. This means that X ranges from 0 to 19 and Y ranges from 0 to 31. The minefield is to occupy the area shown in the diagram.

To lay the mines randomly within the given area we can make use of the RND(N) command. If N is a whole number, then RND(N) returns a whole number between 1 and N. The horizontal co-ordinate of each mine must be chosen to lie

Danger Area
The minefield occupies 20 rows of 16 characters, and the mines are randomly 'sown' during the set-up procedure

between 2 and 17. RND(16) gives numbers from 1 to 16 so RND(16)+1 will select co-ordinates within the area of the field. This procedure will lay as many mines as specified by the value passed to it:

```
2560 DEF PROClay_mines(number_mines)
2570 REM ** CHANGE COLOUR 2 TO GREEN **
2580 VDU19,2,2,0,0,0
2590 FOR I=1 TO number_mines
2600 PRINTTAB(RND(16)+1,RND(25));CHR$(224)
2610 NEXT I
2620 ENDPROC
```

In mode 5 we are restricted to four colours and these are normally black, red, yellow and white, corresponding to colour numbers 0, 1, 2 and 3. However, we don't have to use these colours and can change them using the VDU19 command. The numbers 0 to 3 are known as the *logical* foreground colours. Each of the 16 colours on the BBC has a number that has nothing to do with the mode used. These are called the *actual* colour numbers, a table of which appears on page 224 of the user guide. Any of the four logical colours can be assigned one of the sixteen actual colours. For our game we wish the mines to be green (actual colour number 2). We do not want the yellow that is normally given to us as logical colour 2. The VDU19 command does this.

The detector and assistant take up their initial positions in the bottom-left and top-right corners, respectively. As we will probably want to re-position the detector and assistant later, the procedure for positioning them will use variables (xdet,ydet) for the detector's co-ordinates and (xman,yman) for the assistant's co-ordinates.

```
2830 DEF PROCposition_chars
2840 COLOUR 1
2850 PRINTTAB(xdet,ydet);CHR$(225)
2860 PRINTTAB(xman,yman);CHR$(226)
2870 COLOUR 2
2880 ENDPROC
```

The COLOUR commands at the beginning and end of the procedure select the logical colour that future text will have. COLOUR 1 selects logical colour 1 (red) to print the detector and assistant and COLOUR 2 restores the colour green for future printing. Before this procedure can be used, however, the values of xdet, ydet, xman and yman have to be assigned. This is done in another procedure, along with initialising some other variables that will be used elsewhere.

```
2320 DEF PROCinitialise_variables
2330 xdet=2:ydet=25:xman=17:yman=1
2340 xstart=120:xfinish=1144
2350 zero$="000000"
2360 ENDPROC
```

All these procedures can now be tested by a short calling program as follows:

```
5 MODE 5
10 COLOUR 2
20 PROCinitialise-variables
30 PROCdefine-characters
40 PROClay-mines(40)
50 PROCposition-chars
60 END
```

In the next instalment we shall look at timing and controlling movement from the keyboard.