In addition to the simple functions outlined, the capabilities of the SOUND command can be extended by changing its format to:

```
SOUND &HSFC,V,P,D
```

In this case the ampersand sign & instructs the computer to treat HSFC as a four digit hexadecimal number (see page 179). However, to understand how to use the modified command it is only necessary to analyse the function of each digit. Only three of these are significant as C is merely the usual channel number described in SOUND.

H can either be unspecified (0) or set at 1. H = 1 allows the previous note on the same channel to continue to completion. Otherwise, if a note has been constructed with a long release phase (dying away slowly) the computer will mistakenly assume the note is complete while it is still sounding and start the next note abruptly in the middle of it. H set at 1 forces the computer to wait for completion. If the H parameter is used the remaining parts of the SOUND & command, apart from the channel number, are ignored.

S allows the user to specify a number of notes to be played at the same time on different channels, creating a chord. S = 0 leaves the command in its normal state. S is set to 1 if it is required that a note on one other channel is played at the same time. If set to 2, notes on both the other channels will play. In effect, when the computer encounters an S value it holds the relevant note back until it can account for the associated one or two other notes in the chord that are indicated by the same S number on the remaining channels. It then plays all the specified notes together.

F is set to either 0 or 1. Zero has no effect but 1 causes the computer to discard all notes that are waiting in the channel queue to be played, stops the current note playing and plays the note contained in its own command immediately.

SOUND & is best illustrated with an example. This program plays the first line of 'Happy birthday to you' in the key G# (below middle C) as:

```
G#G#A#G#C#D#
```

It also includes a three part major chord for the final D#

```
10 SOUND 1,-7,37,10:REM *1ST G#*
20 SOUND 1,-7,37,10:REM *2ND SHORT G#*
30 FOR I=1 TO 3: READ N
40 SOUND 1,-7, N,10: NEXT I: REM *A#,G#,C#*
50 SOUND &201, -7,37,15: REM *G#*
60 SOUND &202, -7,53,15: REM *C*
70 SOUND &203,-7,65,15: REM *D#*
80 DATA 45,37,57
90 END
```

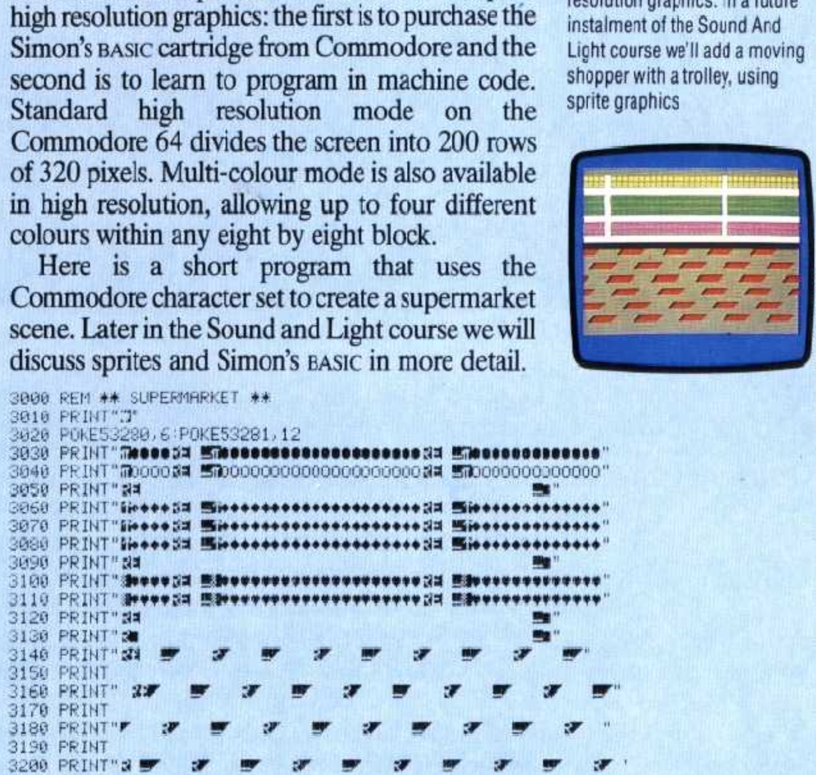
The sound capabilities given by the command ENVELOPE will be discussed later in the course.

Each character cell is made up of eight by eight rows of pixel dots that are represented in binary form as groups of 1s and 0s. Normally 1 is interpreted as a pixel set to the foreground or character colour, and 0 interpreted as a pixel set to the background or screen colour. There is a facility on the Commodore 64 to allow up to four colours to be represented within any character cell. This is known as multi-colour mode. When the computer is switched into multi-colour mode it uses two bits to represent the colour of each pixel.

There are four possible combinations for any pair of bits and this fact is used to represent four different colours inside the character cell. Each character cell on the screen can be set either to be interpreted in the normal way or as a multi-colour character, but in the latter case the choice of colours available is reduced from 16 to 8. Further drawbacks are that the two multi-colours must be the same for all the characters on the screen, and that when in multi-colour mode the horizontal resolution is reduced by a factor of two.

There are no high resolution graphics commands in Commodore BASIC. However, it is possible to create high resolution displays by using the technique known as 'bit-mapping'. The Commodore 64's screen display is made up of 64,000 pixels. Bit-mapping works by allowing the programmer to turn each pixel on or off individually. This is a fairly complex procedure for the average user to undertake and the use of BASIC in this way makes for an extremely slow build-up of the high resolution picture. In practice, there are two alternatives open to those who wish to exploit high resolution graphics: the first is to purchase the Simon's BASIC cartridge from Commodore and the second is to learn to program in machine code. Standard high resolution mode on the Commodore 64 divides the screen into 200 rows of 320 pixels. Multi-colour mode is also available in high resolution, allowing up to four different colours within any eight by eight block.

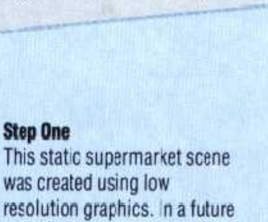
Here is a short program that uses the Commodore character set to create a supermarket scene. Later in the Sound and Light course we will discuss sprites and Simon's BASIC in more detail.



```
3000 REM ** SUPERMARKET **
3010 PRINT"♥"
3020 POKE53280,6:POKE53281,12
3030 PRINT"▮●●●●▮ ▮●●●●●●●●●●●●●●●●●●●●●●▮ ▮●●●●●●●●●●●●●●"
3040 PRINT"▮OOOO▮ ▮OOOOOOOOOOOOOOOOOOOOOO▮ ▮OOOOOOOOOOOOOO"
3050 PRINT"▮                                            ▮"
3060 PRINT"▮♦♦♦♦▮ ▮♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦▮ ▮♦♦♦♦♦♦♦♦♦♦♦♦♦♦"
3070 PRINT"▮♦♦♦♦▮ ▮♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦▮ ▮♦♦♦♦♦♦♦♦♦♦♦♦♦♦"
3080 PRINT"▮♦♦♦♦▮ ▮♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦▮ ▮♦♦♦♦♦♦♦♦♦♦♦♦♦♦"
3090 PRINT"▮                                            ▮"
3100 PRINT"▮♥♥♥♥▮ ▮♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥▮ ▮♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
3110 PRINT"▮♥♥♥♥▮ ▮♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥▮ ▮♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
3120 PRINT"▮                                            ▮"
3130 PRINT"▮                                            ▮"
3140 PRINT"▮  ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮"
3150 PRINT
3160 PRINT" ▮▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮"
3170 PRINT
3180 PRINT"▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   "
3190 PRINT
3200 PRINT"▮ ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮"
3210 PRINT
3220 PRINT"▮  ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮"
3230 PRINT
3240 PRINT" ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮   ▮"
3250 FORI=1063TO2023STEP40
3260 POKEI,160:POKE54272+I,6:NEXT
3270 GOTO3270
```

Step One

This static supermarket scene was created using low resolution graphics. In a future instalment of the Sound And Light course we'll add a moving shopper with a trolley, using sprite graphics



IAN McKINNELL