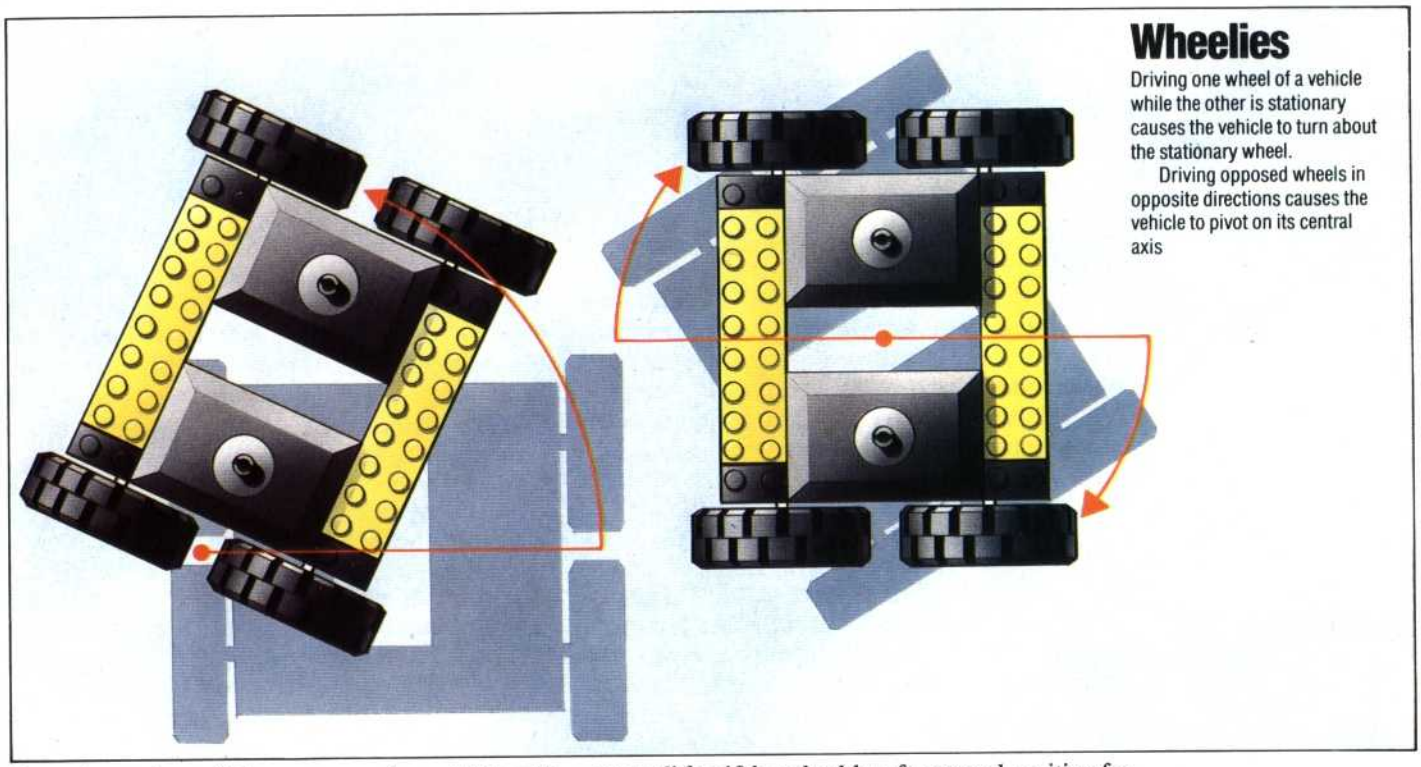

Wheelies

Driving one wheel of a vehicle while the other is stationary causes the vehicle to turn about the stationary wheel.

Driving opposed wheels in opposite directions causes the vehicle to pivot on its central axis

KEVIN JONES

In each version of the program the vehicle will move only while a key is being depressed. As soon as the key is released, the motors are turned off by placing a zero in the data register. The program is exited in each case by pressing the 'X' key.

In the BBC version of the program the procedure TEST-KEYBOARD allows us to test the keyboard directly, rather than reading the keyboard buffer, by using INKEY. This allows more responsive control of the vehicle. The Commodore 64 version firstly turns on the keyboard auto-repeat so that if a key is held down it will keep sending characters into the keyboard buffer to be read by the GET command. Unfortunately there is no way of reading the keyboard directly and accurate control is therefore more difficult than on the BBC Micro. Responsiveness can be improved by clearing out the keyboard buffer just prior to reading it. Inserting the following line into the Commodore version of the program will achieve this.

```
35 GET JS:IF JS<>""THEN35
```

In addition, the GOTO at the end of line 60 should be changed to GOTO35.

The speed at which a key repeats when held down can cause a problem with both versions of this program. If the main program loop is executed faster than the key repeat time, then when the routine comes to test for a keypress again it will think that no key is being pressed. This will result in a rapid switching on and off of the motor as the output alternates rapidly between that set for the chosen direction and zero. In each of the versions of the program this problem has been obviated by adding code to slow down the execution time of the main program loop. In the BBC version using INKEY$(10) causes the computer to 'hang around' for 10 hundredths of a second, waiting for an input, before moving on. In the Commodore 64 version a short delay loop has been added in line 60. The values of these delays were found by a process of trial and error, and are dependent on the length of time needed to execute one pass of the routine. You may find, when writing your own programs, that the routine execution time exceeds the key repeat speed; if not then simply insert a short delay into your code.

Now that we have gained control over the movements of our vehicle it is interesting to design a program that will 'memorise' a sequence of moves and replay them. To do this we can make use of a two-dimensional array that records direction and the time taken for each different manoeuvre made. The first part of such a program will be the same as those already given but the second part will replay the stored data. The data will be stored in an array DR(), where DR(C,1) stores direction and DR(C,2) stores the time taken for each movement. A new element in the array is used each time a new direction is selected. This condition is indicated by a change in the contents of the data register. A counter, C, is used to keep track of the array elements.

BBC MICRO

```
1000 REM BBC MOVEMENT MEMORY
1010 DDR=&FE62:DATREG=&FE60
1020 DIM DR(100,2)
1030 ?DDR=255:C=1:REM INIT COUNT
1040 REPEAT
1050 A$=INKEY$(10)
1060 PROCtest_keyboard
1070 UNTILA$="X"
1080 ?DATREG=0
1090 DR(C-1,2)=TIME
1100 REPEAT A$=GET$
1110 UNTIL A$="C"
```