(lines 500 to 570 are the keyboard routine that sets new values for a, b, i and j)

```
600 LET x=x+a: LET y=y+b
610 LET m=m+i: LET n=n+j
620 GOTO 400
```

The only confusing point is perhaps line 410, where the variables for bike one are set to those for bike two, and a new variable,col, is introduced. This is so that a single routine can be used for the collision action, where x and y are simply used to indicate the point at which the collision takes place, and col sets the colour.

The routine for checking the keyboard has to be fast, but we regretfully have to use IF . . . THEN statements that are fairly slow. However, we can use the fast IN command to read the keys. The control keys chosen are Q and A, which control upward and downward movement for bike one, and P and ENTER for bike two. Left and right are X and C for bike one, and N and M for bike two. (See page 366 for a full explanation of how blocks of keys relate to bytes in memory.)

```
500 IF IN 64510= 190 THEN LET a=0: LET b=1
510 IF IN 65022= 190 THEN LET a=0: LET b= -1
520 IF IN 65278= 187 THEN LET a= -1: LET b=0
530 IF IN 65278= 183 THEN LET a=1: LET b=0
540 IF IN 57342= 190 THEN LET i=0: LET j=1
550 IF IN 49150= 190 THEN LET i=0: LET j= -1
560 IF IN 32766= 187 THEN LET i=1: LET j=0
570 IF IN 32766= 183 THEN LET i= -1:LET j=0
```

All that remains is the collision routine, and the updating of the scores. An expanding series of concentric circles, centred on the point of impact was chosen, with radii of four, six and eight pixels:

```
700 FOR d=1 TO 3
710 CIRCLE BRIGHT 1; INK col; x,y,2+d*2
720 NEXT d
730 IF col=6 THEN LET p=p+1: GOTO 750
740 LET q=q+1
750 GOTO 100
```

This finishes the game, with the last statement looping back to the initialising procedures at the beginning. The game could, however, do with a starting procedure to make it more friendly to use:

```
300 PRINT AT 10,5; INK 7; "PRESS ANY KEY TO
   START"
310 IF INKEY$= "" THEN GOTO 310
320 PRINT AT 10,5; "                        "
```

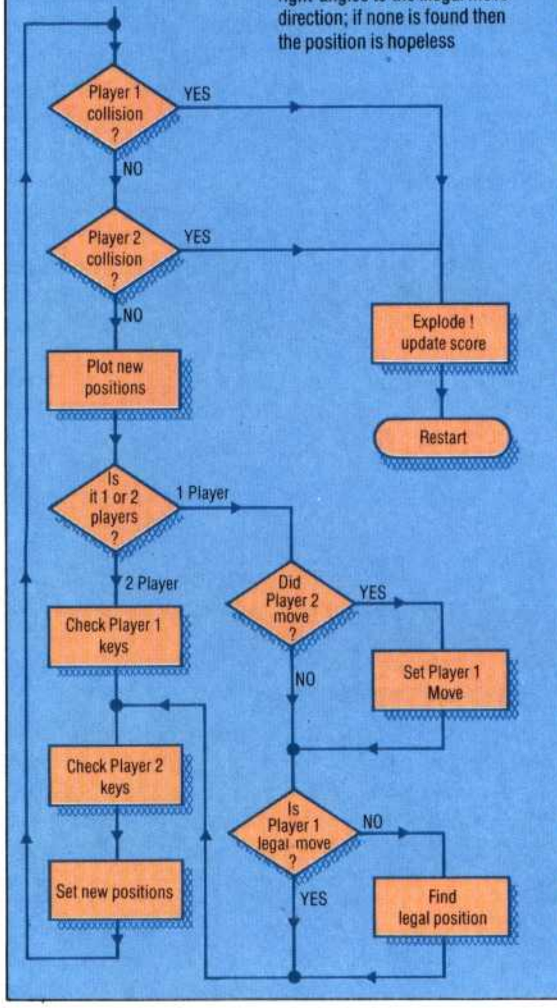
This gives you a break between consecutive rounds. All that remains is to save the game to cassette, preferably using SAVE "OnYerBike" LINE 10 so that the game automatically runs as soon as it is loaded.

The game could obviously be made more exciting, with instruction screens, loading screens, a one-player option with a strategy routine controlling the other bike as we have suggested, sound and better graphics. But certainly the latter options would make the game unplayably slow. In a future instalment we will rewrite this game in machine code to demonstrate its full potental.

One Hand Clapping

Fitting In

In the one-player version the computer is Player 1. The program section that checks Player 1's command keys for input is bypassed, and the algorithm in the code given allows Player 1 to continue moving in a straight line until that generates an illegal move, or Player 2 moves. In the latter case, this move is mirrored or duplicated, and then checked for legality. Whenever an illegal Player 1 move is generated, the program looks for legal moves at right-angles to the illegal move direction; if none is found then the position is hopeless



There are many ways of implementing a one-player version of this game. The changes we suggest make it possible to choose between the one- and two-player versions at the start of every game. It isn't difficult to invent satisfactory algorithms for playing this game, but it's extremely difficult to implement them in BASIC without slowing down the game considerably. Make the following changes to the program:

```
20 LET keybd=500:LET
   pt=-1:LET umoved=0:
   RANDOMIZE
260 PRINT AT 10,5;"No. of
   Players (1/2) ?"
270 LET a$=INKEY$:IF
   a$<>"1" AND a$<>"2"
   THEN GO TO 270
280 IF INKEY$<>"2" THEN
   LET keybd=440
```

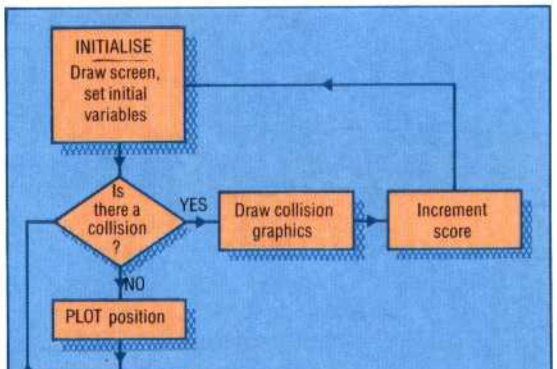
These lines give the user the choice of game types, and implement that choice by accessing either the one-player stratagem between lines 440 and 460, or the standard two-player version between lines 500 and 570. Our strategy is contained in these lines:

```
430 GOTO keybd
440 LET pt=SGN (RND-0.5): IF
   umoved=pt THEN LET a=pt*j:
   LET b=pt*i: LET umoved=0
450 IF POINT (x+a,y+b)<>1
   THEN GOTO 540
460 LET pt=SGN (RND-.5): LET
   a=a*pt: LET b=b*pt: LET d=a:
   LET a=b: LET b=d: IF POINT
   (x+a,y+b)<>1 THEN GOTO
   540
490 LET a=-a: LET b=-b: GOTO
   540
540 IF IN 57342=190 THEN LET
i=0: LET j=1: LET umoved=1
```

and similarly add:

```
: LET umoved=1
```

to the end of lines 550 to 570



One-Player Option

This simplified flowchart shows the program structure with only one player. Each process is repeated in the full two-player game