10 MIGLIA

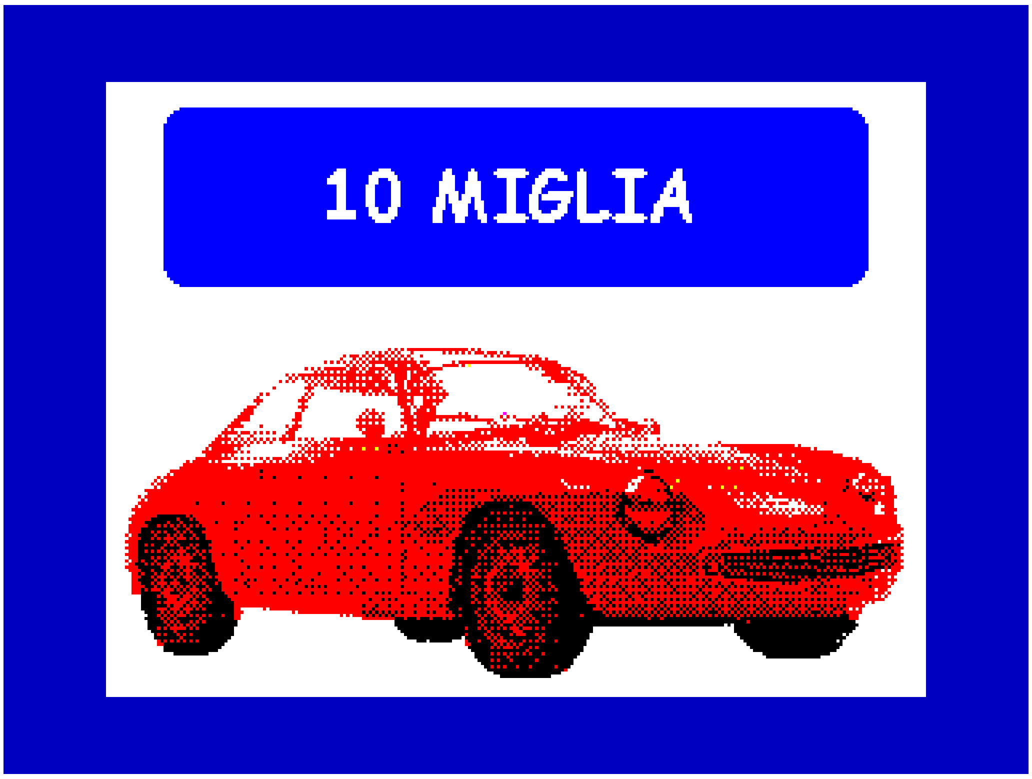
BASIC 10 Liner car racing/driving game for the Sinclair ZX Spectrum

Marco Varesio (Marco's Retrobits)

English language blog

Italian language blog

YouTube channel

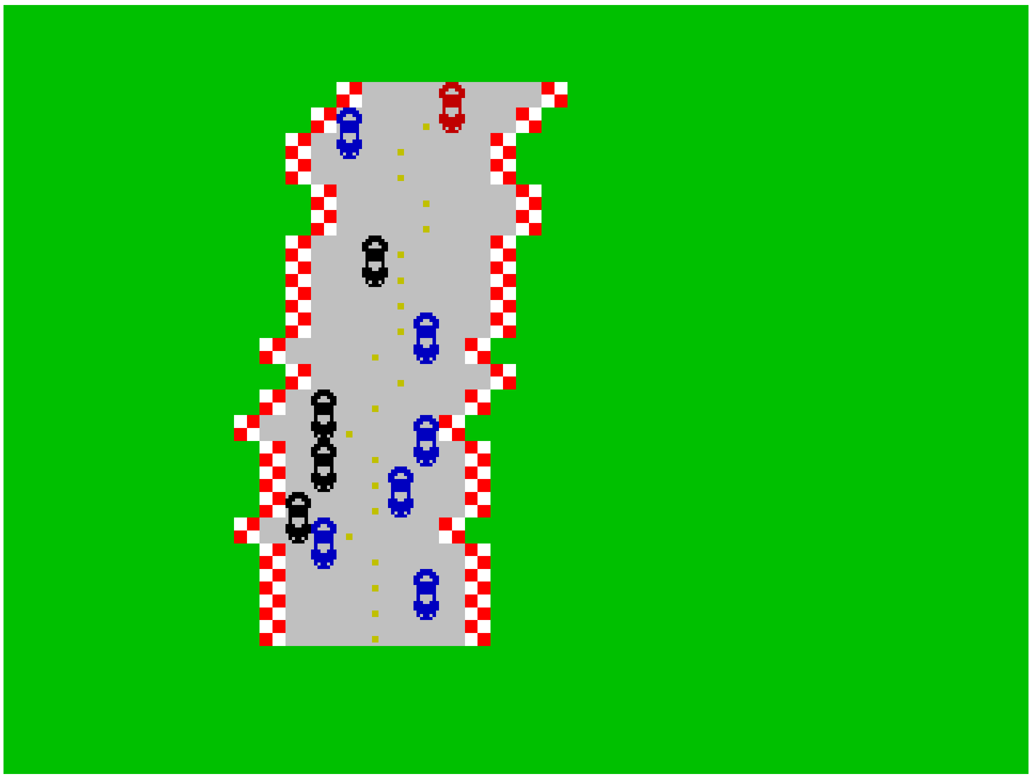


Warm up your car's engine for the most exciting 10-liner race in the 8-bit computer world! Three, two, one... GO!

10 Miglia is a BASIC driving game for the Sinclair ZX Spectrum home computer. It is my entry to the 2021 edition of the BASIC 10 Liner Contest , PUR-80 category. This means that the program is made up of 10 lines of BASIC code (max 80 characters per logical line).

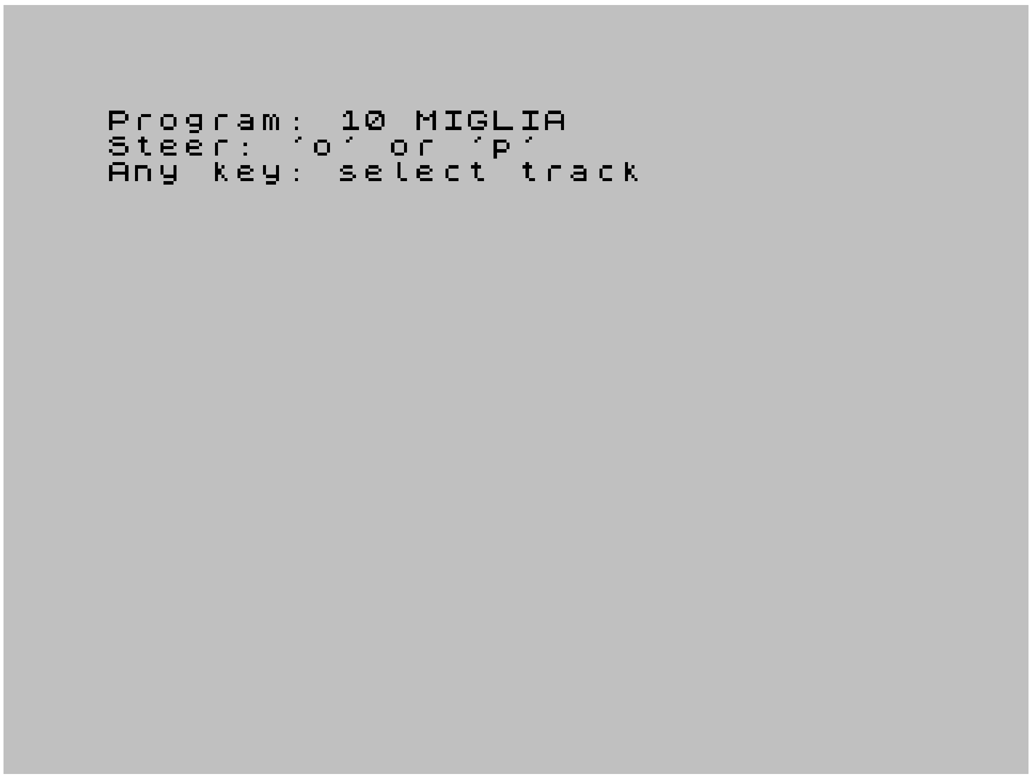
Select your favourite track and drive your shiny red vintage car, avoiding accidents with other cars and going off the road. Achieve the highest score by going as far as you can.

Controls: steer your car with "o" (left) and "p" (right).



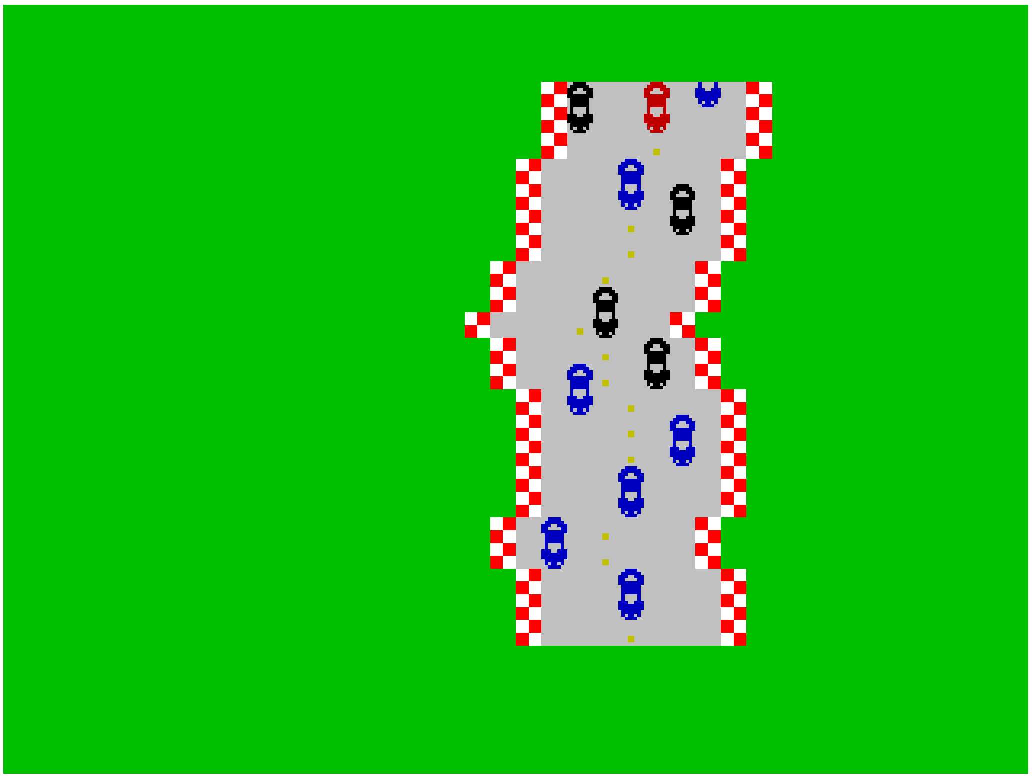
Game description

When the program starts, it informs you about the controls and prompts you to select your desired track, by tapping any key.

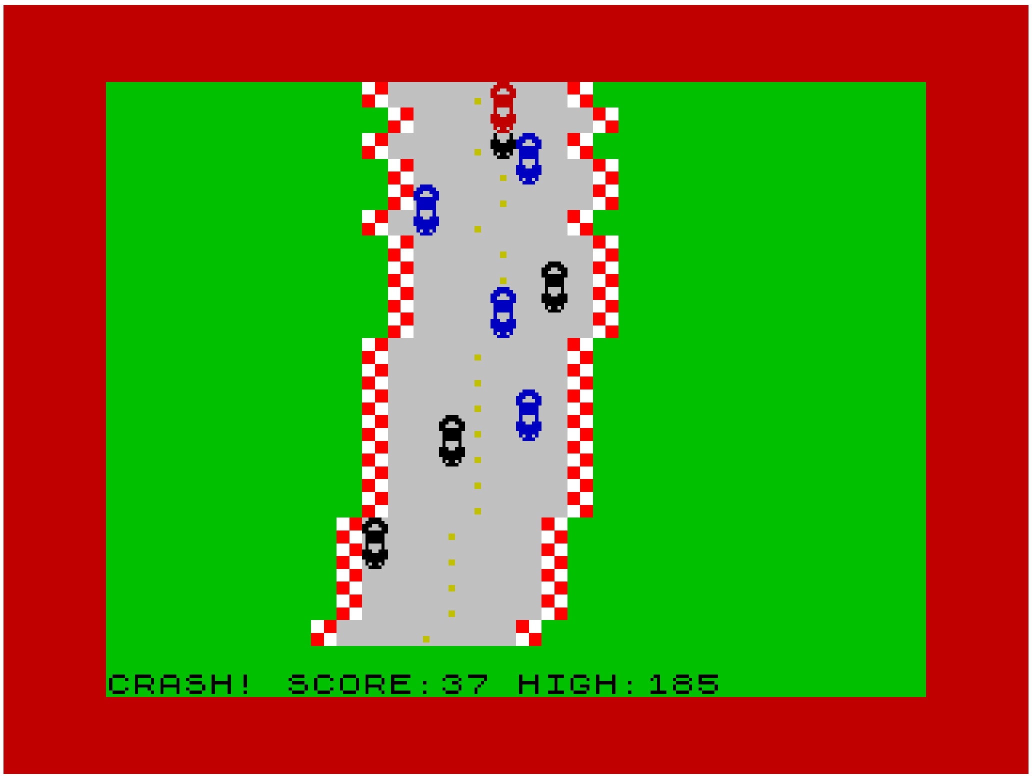


```
Program: 10 MIGLIA
Steer: 'o' or 'p'
Any key: select track
```

After selecting the track, the game starts. Your car is the red one in the top line of the screen and heads towards the bottom. Use the "o" and "p" keys to steer your car and avoid crashes.



The highest score is recorded during the gaming session. The game is over either when your car hits another car or when it goes off the road. When the game is over, both your current score and the highest score are shown.



```
CRASH!  SCORE:37  HIGH:185
```

The game will restart with selected track by hitting any key. Reload the program to play a different track; it's only 10 lines, so loading is fast! ;-)

Program description

Variables

Variable name	Purpose
H	High score
S	Score
D	Random seed
A	Random number
M	Screen permanent attributes **system variable** memory address
I	Loop iterator; road segment Y position
R	Road segment left side X position
B	Other cars ink colour, alternating between 0 (BLACK) and 1 (BLUE)
C	Other car X position
P	Previous other car X position (used to avoid car overlapping)
X	Player car x position
Z	Screen attribute of the next car fron position; used for collision detection
A$	Crash (game over) message
B$	Single space string; used to "erase" the car from its previous position
E$	Car rear **user-defined graphics (UDG)** character (code 90 HEX)
F$	Car front **user-defined graphics (UDG)** character (code 91 HEX)
K$	User input pressed key

Program listing

```
1 LET M=23693:LET H=0:PRINT"Steer: 'o' or 'p'","Any key: select track":PAUSE 0:LET
D=CODE INKEY$:LET B$=" ":RANDOMIZE D
2 LET E$="{90}":LET F$="{91}":RESTORE 10:FOR I=0TO15:READ A:POKE 65368+I,A:NEXT
I:BORDER 4:POKE M,34:CLS:READ A$:LET R=11:LET X=R+2
3 FOR I=21TO0 STEP -1:GO SUB 9:BEEP .2*(I=0)+.1*(INT(I/6)=I/6),20+5*(I=0):NEXT I:LET
I=21:LET B=0:LET P=0:LET S=0
4 POKE M,62:PRINT AT 0,X;B$;AT 1,X;B$:POKE M,34:LET K$=INKEY$:LET A=USR 3582:GO SUB
9:LET A=RND:LET X=X+(K$="p")-(K$="o")
5 LET Z=PEEK(22560+X):POKE M,58:PRINT AT 0,X;E$;AT 1,X;F$:LET R=R+(R<20)*RND-
(R>2)*A:LET A=A*10
6 IF A<6 THEN POKE M,56+B:LET C=INT(R+1+A):IF C<>P THEN LET P=C:PRINT AT 20,C;E$;AT
21,C;F$
7 IF Z<>62 THEN LET H=(H>S)*H+(H<=S)*S:PRINT#1;A$;S;" HIGH:";H:BORDER 2:BEEP
.5,-15:RANDOMIZE D:PAUSE 0:GO TO 2
8 LET S=S+1:LET B=(B=0):GO TO 4:REM MARCO V. 2021 MARCO'S RETROBITS
https://retrobits.itch.io
9 POKE M,87:PRINT AT I,R;"{86}";:POKE M,62:PRINT"   .   ";:POKE M,87:PRINT"{89}"
10 RETURN:DATA 60,126,231,195,255,126,126,126,66,66,231,255,255,126,90,60,"CRASH!
SCORE:"
```

Please note that this program listing is formatted in order to be used with the bas2tap utility, which converts a BASIC listing in an ASCII file to a .TAP emulator tape image. In particular, the ZX Spectrum ASCII character values are represented as hexadecimal values in curly braces. For example, the characters that represent the sides of the road are "{86}" and "{89}". If you want to type this listing (on a real or emulated machine) rather than loading the provided tape image, you must replace these values with the corresponding characters. The same applies to the car user defined graphic charcters, with hex codes 90 and 91.

Source code explained

Line 1 is executed only once when the program is loaded; it performs some initializations and prompts the player to choose the desired track by pressing any key. Tracks are (pseudo) randomly generated; however, the code of the character associated to the pressed key will be used as seed for the random generator. In this way, each track will always look the same, even across different plays.

```
1 LET M=23693:LET H=0:PRINT"Steer: 'o' or 'p'","Any key: select track":PAUSE 0:LET
D=CODE INKEY$:LET B$=" ":RANDOMIZE D
```

The first statement stores in M the address of the system variable ATTR_P, containing the screen INK (foreground colour) and PAPER (background colour) attributes. This allows to set these values by writing to this memory location, thus using only one POKE instruction instead of the INK, PAPER pair. The second LET statement initializes the high score H to 0.
Then, the program prints some instructions and waits for a key press. The code of the corresponding character is stored in D; this value is finally used to initialize the pseudorandom generator. B$ is initialized to the space character, and will be used to "erase" the player's car from its previous position, before drawing it to the new position at each game loop.

Line 2 performs some other initializations and displays the road. It is called every time a new game starts:

```
2 LET E$="{90}":LET F$="{91}":RESTORE 10:FOR I=0TO15:READ A:POKE 65368+I,A:NEXT
I:BORDER 4:POKE M,34:CLS:READ A$:LET R=11:LET X=R+2
```

E$ and F$ are assigned to the user-defined graphic characters corresponding to the rear and front of the cars; then, the FOR loop loads the shape of these graphics, by READing them from line 10. The BORDER colour is set to green (4) and the POKE M, 34 statement is used to set the PAPER colour to green (4) and INK colour to red (2); see the screen memory layout documentation for details. The CLS statement clears the screen by setting it to the previously set PAPER colour (green). A$ is initialized to the game over message, read from line 10. The road left edge position R is set to 11 and the player's car horizontal position is set 2 characters right to the left edge of the road.

Line 3 draws the road and performs other initializations:

```
3 FOR I=21TO0 STEP -1:GO SUB 9:BEEP .2*(I=0)+.1*(INT(I/6)=I/6),20+5*(I=0):NEXT I:LET
I=21:LET B=0:LET P=0:LET S=0
```

The for loop is used to draw the road, by calling 22 times (one for each screen row, from bottom to top) the routine that draws a road segment, located at line 9. While the road is being displayed, the BEEP statement is executed 4 times (when the iterator value I is a multiple of 6). The sounds are used to warn the user that the race is starting (Thre, Two, One, GO!, you know). The last beep (when I=0) is longer and has a higher pitch. Current score S is initialized to 0, as well as helper variables B and P.

The actual game loop starts at line 4:

```
4 POKE M,62:PRINT AT 0,X;B$;AT 1,X;B$:POKE M,34:LET K$=INKEY$:LET A=USR 3582:GO SUB
9:LET A=RND:LET X=X+(K$="p")-(K$="o")
```

The first POKE statement sets the screen attributes to the road colours (PAPER 7, which corresponds to white - actually grey - for the asphalt and INK 6, which represents yellow, for the road line). In this way, the effect of the subsequent PRINTs of white spaces at the player's car position is to "erase" the player's car. The next POKE sets again the screen attributes to green paper, so that when the screen is scrolled up, the new line added at the bottom will be green.
The USR statement implements a call to the scrolling routine, located in the ZX Spectrum ROM at address 3582 (HEX: 0DFE), which scrolls the screen up by one character line. The new empty line at the bottom of the screen is filled with a new row segment, by calling the subroutine at line 9.
A new random number, that will be used later, is generated and stored into A. Then, if either the "o" or the "p" key is pressed, the variable storing the new X position of the player's car is incremented or decremented by 1 respectively. In fact, if the pressed key is "o", (K$="p") evaluates to 0 and (K$="o") evaluates to 1, so X is decremented by 1: (K$="p")-(K$="o") = 0-1 = -1. Conversely, if the pressed key is "p", the expression (K$="p")-(K$="o") evaluates to 1-0 = 1 and X is incremented by 1.

Line 5 prepares for collision detection, draws the player's car at the new calculated position and determines the next row segment horizontal position:

```
5 LET Z=PEEK(22560+X):POKE M,58:PRINT AT 0,X;E$;AT 1,X;F$:LET R=R+(R<20)*RND-
(R>2)*A:LET A=A*10
```

The PEEK statement reads the screen attributes of the new position of the player's car front. This information will be used for collision detection. The POKE M, 58 statement sets the screen attributes to the player's car colours, i.e. red (2) INK on white/grey paper (7). Now we can print the player's car at the new X position in the top lines of the screen. After that, the horizontal displacement of the next road segment to be printed R is randomly determined: it will be either the same as current one or on character left or one character right. The program also ensures that the road segment stays within the screen, i.e. the left side of the road stays within columns 2 and 20. Finally, A is multiplied by 10 on order to have a random number between 0 and 10.

Line 6 draws a new opponent car:

```
6 IF A<6 THEN POKE M,56+B:LET C=INT(R+1+A):IF C<>P THEN LET P=C:PRINT AT 20,C;E$;AT
21,C;F$
```

With a probability of a bit less than 60% (A<6), the program draws a new opponent car at the bottom of the screen. The opponent car INK colour will be either black (0) or blue (1) based on the value of B, while the PAPER colour is white/grey. The opponent car horizontal position C is randomly calculated within the sides of the road and the car is printed only if its position C is different from the previously added opponent car position P. The newly calculated C value is saved into P.

Line 7 handles collision with either another car or one of the sides of the road:

```
7 IF Z<>62 THEN LET H=(H>S)*H+(H<=S)*S:PRINT#1;A$;S;" HIGH:";H:BORDER 2:BEEP
.5,-15:RANDOMIZE D:PAUSE 0:GO TO 2
```

From the PEEK statement in line 5, the Z variable holds the colour values of the position the player's car is heading to. So, if this value corresponds to the empty road colours (yellow INK on white/grey PAPER), there is no collision; otherwise, the player's car is either over the edge of the road or over another car. If so, high score H is updated to the maximum between previous high score and current score S and then the "CRASH!" game over message is printed. The border colour is set to red (2) and a sad tone is BEEPed. Finally, the program reinitializes the random generator and waits for any key press before starting a new game, by jumping to line 2.

If the game is not over, the program executes line 8:

```
8 LET S=S+1:LET B=(B=0):GO TO 4:REM MARCO V. 2021 MARCO'S RETROBITS
https://retrobits.itch.io
```

Current score is incremented by one and the INK colour of the next opponent car to be added is updated: if the pevious car was black, the next car will be blue and vice-versa. The program goes to line 4 and the game loop restarts.

Line 9 contains the subroutine that prints a road segment:

```
9 POKE M,87:PRINT AT I,R;"{86}";:POKE M,62:PRINT"   .   ";:POKE M,87:PRINT"{89}"
```

The POKE M,87 statement is used to set the colour of the road edges (bright red INK on bright white PAPER), while 62 is the usual attributes value for the road itself. The edges are displayed by means of the chequered characters with codes 86 HEX and 89 HEX, while the road is nothing else but 6 spaces with a full stop in the middle. The road segment is printed at row I, column R.

The last line contains the RETURN statement for returning from the subroutine starting at the previous line and DATA for the cars graphics and game over message:

```
10 RETURN:DATA 60,126,231,195,255,126,126,126,66,66,231,255,255,126,90,60,"CRASH!
SCORE:"
```

Program lines length proof

By replacing each BASIC token with a single character and removing redundant blank spaces, each line in the resulting source code does not exceed the 80 characters limit:

```
1lM=23693:lH=0:p"Steer:'o' or 'p'","Any key:select track":m0:lD=IN:lB$=" ":tD
2lE$="{90}":lF$="{91}":S10:fI=0F15:AA:o65368+I,A:nI:b4:oM,34:v:AA$:lR=11:lX=R+2
3fI=21F0 D-1:h9:Z.2*(I=0)+.1*(INT(I/6)=I/6),20+5*(I=0):nI:lI=21:lB=0:lP=0:lS=0
4oM,62:pI0,X;B$;I1,X;B$:oM,34:lK$=N:lA=L3582:h9:lA=T:lX=X+(K$="p")-(K$="o")
5lZ=O(22560+X):oM,58:pI0,X;E$;I1,X;F$:lR=R+(R<20)*T-(R>2)*A:lA=A*10
6uA<6 GoM,56+B:lC=INT(R+1+A):uC<>P GlP=C:pI20,C;E$;I21,C;F$
7uZ<>62 GlH=(H>S)*H+(H<=S)*S:p#1;A$;S;" HIGH:";H:b2:Z.5,-15:tD:m0:g2
8lS=S+1:lB=(B=0):g4:eMARCO V. 2021 MARCO'S RETROBITS https://retrobits.itch.io
9oM,87:pII,R;"{86}";:oM,62:p"   .   ";:oM,87:p"{89}"
10y:D60,126,231,195,255,126,126,126,66,66,231,255,255,126,90,60,"CRASH! SCORE:"
```

Considering that the above listing contains graphic character hexadecimal codes instead of the actual graphic characters, the length of lines 2 and 9 could be further reduced.

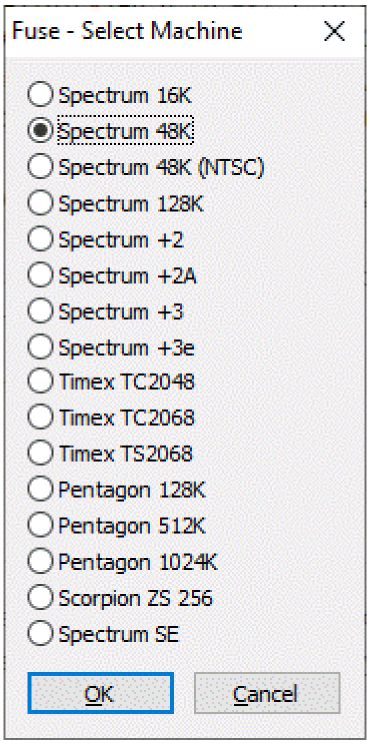
Therefore, 10 Miglia is a suitable entry for the PUR-80 category of the BASIC 10 Liner Contest.

Loading instructions

10 Miglia is provided in TAP tape image format, which can be easily loaded in most ZX Spectrum emulators and on the real machines, either equipped with devices such as the DivMMC or by playing it through the MIC port using tools like PlayTZX or WinTZX.

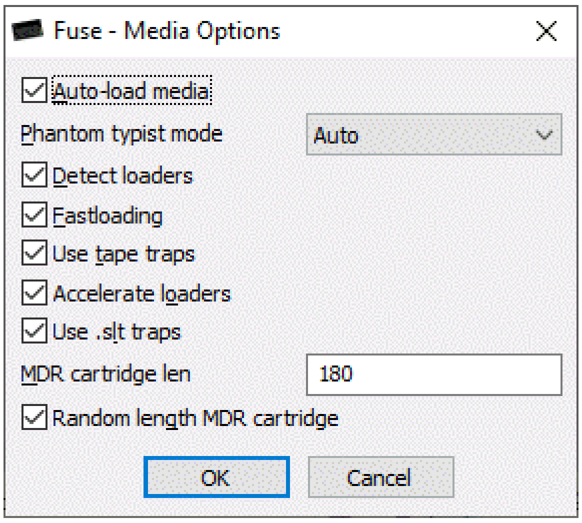
The following instructions apply to the Fuse open source emulator, which is available for Unix, Linux, Windows, macOS and many other platforms. For other emulators or devices, please refer to their specific documentation for loading TAP files.

Start the Fuse emulator and select the Spectrum 48K model in "Machine" -> "Select..."


Fuse - Select Machine

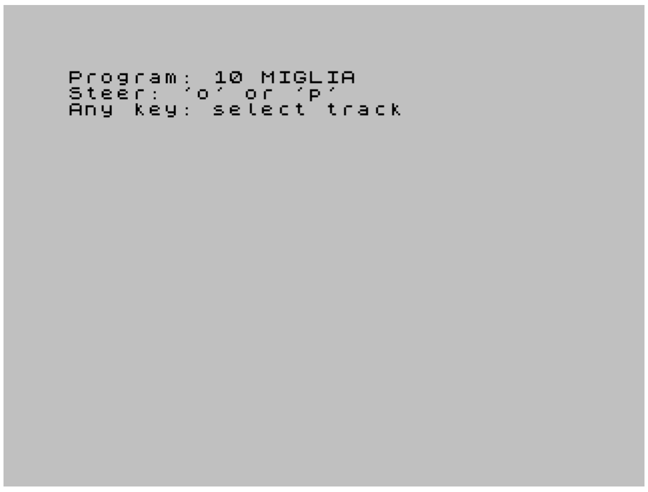
- Spectrum 16K
- [✓ Spectrum 48K]
- Spectrum 48K (NTSC)
- Spectrum 128K
- Spectrum +2
- Spectrum +2A
- Spectrum +3
- Spectrum +3e
- Timex TC2048
- Timex TC2068
- Timex TS2068
- Pentagon 128K
- Pentagon 512K
- Pentagon 1024K
- Scorpion ZS 256
- Spectrum SE

Make sure that automatic loading of tape image files is enabled, by checking the corresponding options in "Options" -> "Media..."


Fuse - Media Options

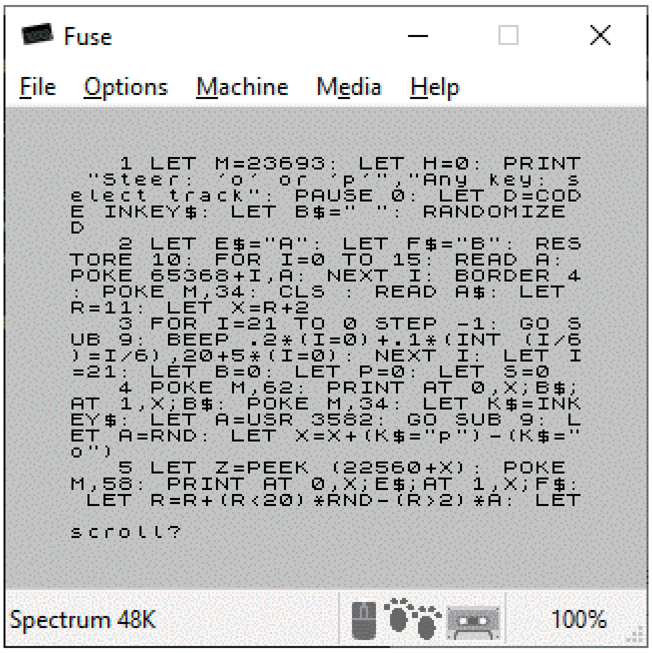
- ✓ Auto-load media
- Phantom typist mode: Auto
- ✓ Detect loaders
- ✓ Fastloading
- ✓ Use tape traps
- ✓ Accelerate loaders
- ✓ Use .slt traps
- MDR cartridge len: 180
- ✓ Random length MDR cartridge

Open the **10miglia.tap** file, either by selecting it in "File"->"Open..." or by dragging and dropping it on the emulator window.



```
Program: 10 MIGLIA
Steer: 'O' or 'P'
Any key: select track
```

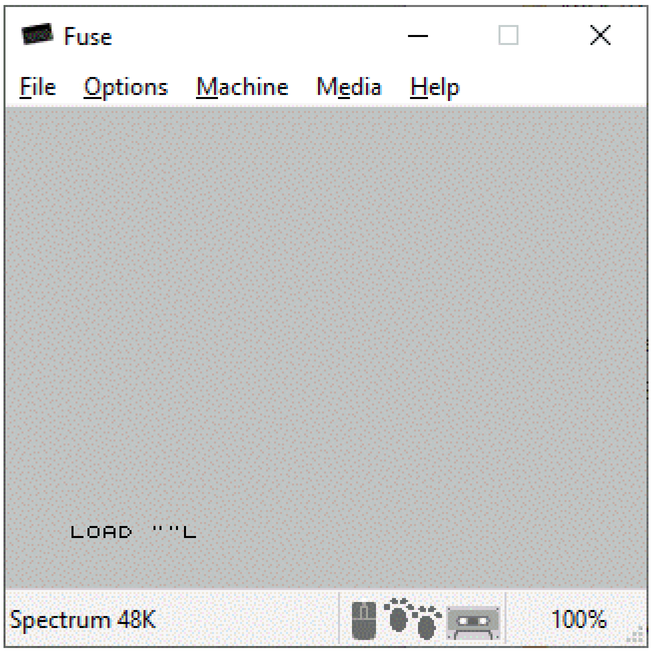
To see the program listing, BREAK the program by pressing SHIFT + SPACE BAR (the CAPS SHIFT ZX Spectrum key is usually mapped to SHIFT). The "L BREAK into program" message will be displayed. Then press the K key, followed by the ENTER key. Now you will see the first page of the program listing; to scroll to the next page, press any key except SPACE BAR.



```
   1 LET M=23693: LET H=0: PRINT
 "Steer: 'o' or 'p'","Any key: s
elect track": PAUSE 0: LET D=COD
E INKEY$: LET B$=" ": RANDOMIZE
D
   2 LET E$="A": LET F$="B": RES
TORE 10: FOR I=0 TO 15: READ A:
POKE 65368+I,A: NEXT I: BORDER 4
: POKE M,34: CLS : READ A$: LET
R=11: LET X=R+2
   3 FOR I=21 TO 0 STEP -1: GO S
UB 9: BEEP .2*(I=0)+.1*(INT (I/6
)=I/6),20+5*(I=0): NEXT I: LET I
=21: LET B=0: LET P=0: LET S=0
   4 POKE M,62: PRINT AT 0,X;B$;
AT 1,X;B$: POKE M,34: LET K$=INK
EY$: LET A=USR 3582: GO SUB 9: L
ET A=RND: LET X=X+(K$="p")-(K$="
o")
   5 LET Z=PEEK (22560+X): POKE
M,58: PRINT AT 0,X;E$;AT 1,X;F$:
 LET R=R+(R<20)*RND-(R>2)*A: LET

scroll?
```

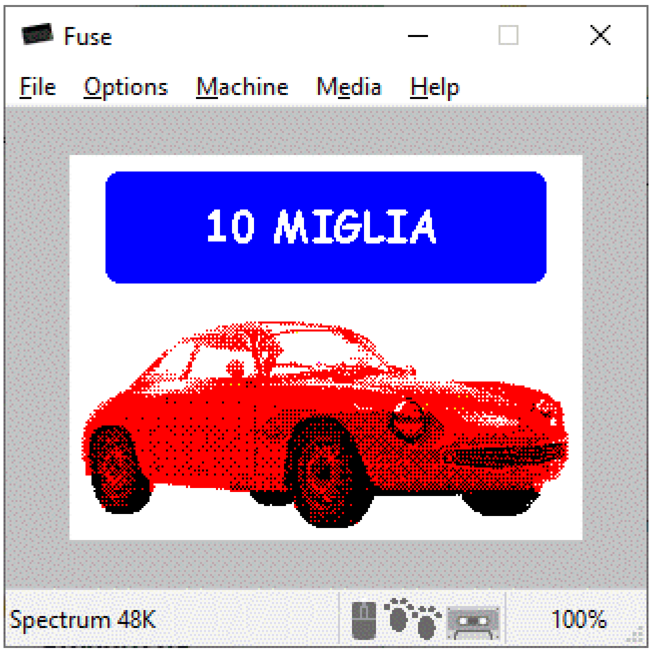
If your emulator of choice does not support automatic tape loading, after mounting the tape image you must manually issue the tape loading command, by pressing the J key, followed by CTRL + P twice (the SYMBOL SHIFT ZX Spectrum key is usually mapped to CTRL) and then by ENTER.



```
LOAD ""L
```

If you are emulating a 128K ZX Spectrum model, to start loading simply select "Tape Loader" using the cursor keys in the main system menu and press ENTER.

A bonus artwork tape image file with the game logo (**10Miglia_BonusArtwork.tap**) is also included. To prevent the "OK" message from breaking the image, use the following command to load and wait for a key press after loading: `LOAD "" SCREEN$ : PAUSE 0`


Fuse
File
Options
Machine
Media
Help
10 MIGLIA
Spectrum 48K
100%