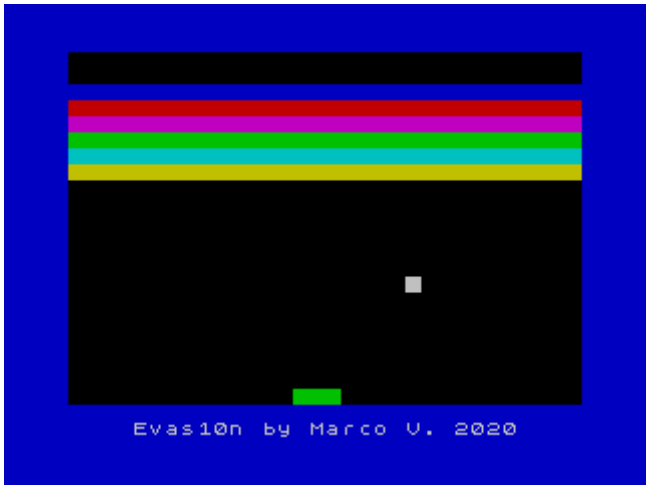
Evas10n

10-Liner BASIC Breakout game for the Sinclair ZX Spectrum

Marco Varesio
https://retrobits.altervista.org/
https://retrobits.itch.io/
https://somebitsofme.altervista.org/



Evas10n is a Breakout – like videogame for the Sinclair ZX Spectrum home computer, implemented in 10 lines of BASIC (max 80 characters per line). Differently from most similar games, in Evas10n you do not have to destroy all the walls, but you must get through the walls to the top of the screen. In this aspect, it is similar to CHIP-8 Breakout implementation by Carmelo Cortez (which you can play online using my CHIP-8 virtual machine), derived from Wipe Off by Joseph Weisbecker.

Control the paddle with "z" (left) and "x" (right) keys and bounce the "ball" against the bricks, making your way to the top of the screen. When the ball hits the paddle surface in the middle, its horizontal direction is unchanged; instead, if the hit point is one of the paddle surface corners, the ball direction is inverted. If the ball touches the bottom of the screen, it is lost. You have 6 balls available. The game is completed when the ball reaches the top of the screen. If you succeed, the "FREE!" message will be shown, and the game will freeze. If you fail, the game will automatically restart.

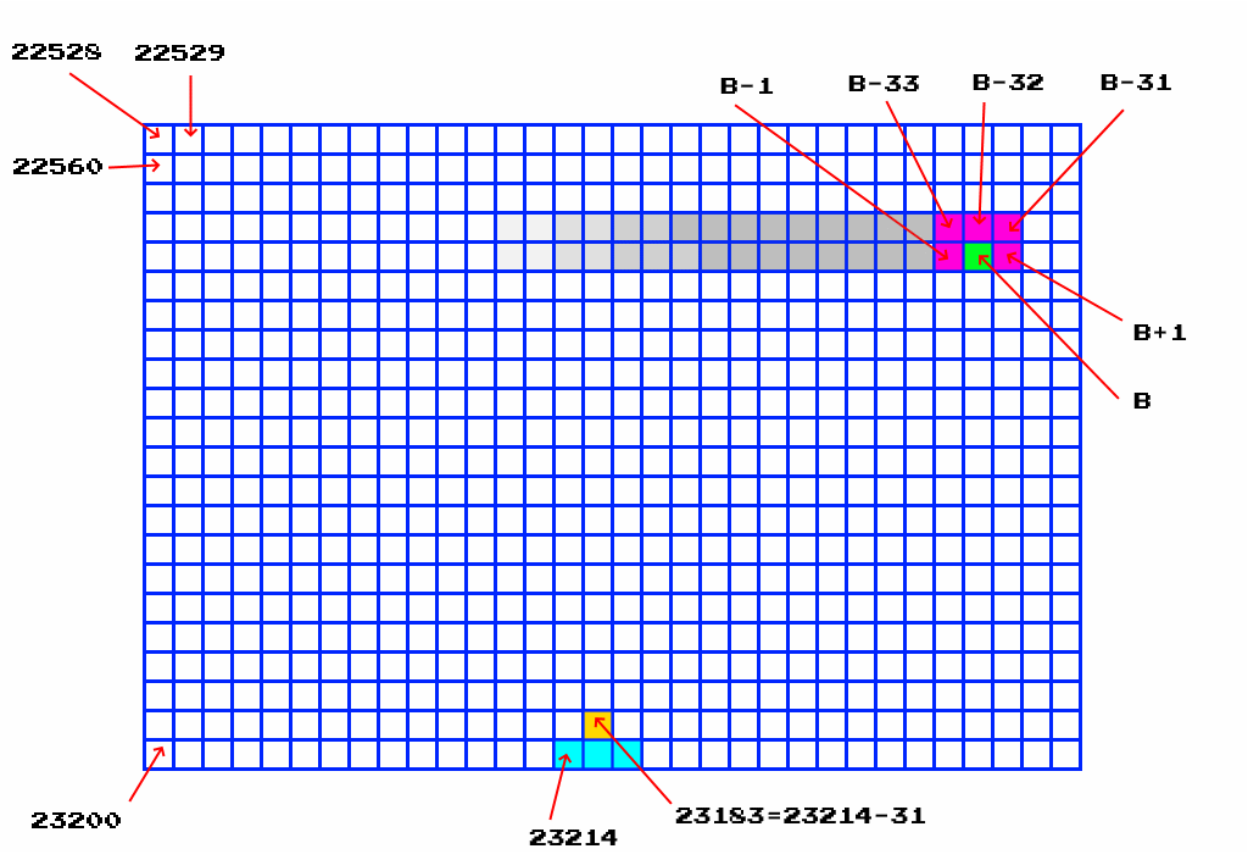
Program description

The game is my entry to the 2020 edition of BASIC 10-Liner contest, "PUR-80" category. Even if I could take advantage of the fact that ZX Spectrum BASIC tokens count as 1 character and spaces count as 0 characters, and so having an actual textual listing with lines longer than 80 characters, I liked the minimalistic result I achieved with all text lines <= 80 character and so, at the cost of cutting out some functionality, I left the code that way.

```
1 BORDER 1:INK 0:PAPER 0:CLS:LET L=6:LET A=23214:LET V=1:FOR I=32 TO 223
2 POKE 22560+I,8*INT(I/32):NEXT I:PRINT #1;AT 1,4;"Evas10n by Marco V. 2020"
3 LET W=-1:LET X=A-23199:LET Y=20:LET B=A-31:FOR I=0 TO 2:POKE A+I,32:NEXT I
4 LET X=X+V:IF(INKEY$="x")*(A<23229)THEN POKE A+3,32:POKE A,0:LET A=A+1
5 LET Y=Y+W:IF(INKEY$="z")*(A>23200)THEN LET A=A-1:POKE A,32:POKE A+3,0
6 POKE B,0:LET B=B+V+32*W:IF Y=21 THEN BEEP .7,-23:LET L=L-1:GO TO 2*(L>0)+1
7 IF PEEK(B+V) AND X>0 AND X<31 THEN POKE B+V,0:LET V=-V:LET W=1:GO TO 9
8 POKE B,56:LET D=B-A+31:LET E=ABS D:IF PEEK (B-32) THEN POKE (B-32), 0:LET W=1
9 LET W=-(E<3)+(E>2)*W:IF Y=0 THEN PRINT #1;AT 0,0;"FREE!":BORDER 4:GO TO 9
10 LET V=(E<>2)*V+(D=2)-(D=-2):LET V=(X=0)-(X=31)+V*(X<>0 AND X<>31):GO TO 4
```

Graphics and screen memory access

Rather than using slow PRINT statements to draw things on the screen, I adopted the faster method of POKEing the screen attributes memory area (offset: 22528, length: 768 bytes) for coloring each character position. Since the ZX Spectrum resolution is 256*192, there are 32*24 character positions (only 22 lines are used for the game, however, since the last 2 lines are managed differently in BASIC), each character being 8*8 pixels. The screen is initialized with black (ink=0) space characters on black background (paper=0). Then, for "turning on" a character position, I used the POKE statement with a value different than 0 to set its background (paper) color, as depicted below:



For more details on the ZX Spectrum screen memory layout, please refer to:
http://www.breakintoprogram.co.uk/computers/zx-spectrum/screen-memory-layout and
http://www.worldofspectrum.org/ZXBasicManual/zxmanchap24.html

Variables

A = memory address of the leftmost paddle character

B = memory address of the ball character

X = ball x coordinate

Y = ball y coordinate

V = horizontal ball direction (-1=right to left; 1=left to right)

W = vertical ball direction (1=down; -1=up)

D = distance (in characters) between the paddle and the ball (based on difference between B and A)

E = absolute value of D

I = generic iterator

L = balls left

Source code explained line by line

Game initialization: set screen outer BORDER color to blue (1) and clear the screen setting both foreground (INK) and background (PAPER) color to black (0). LET L=10 makes you start with 6 balls. Set A to the address of the paddle leftmost character attribute, then initialize horizontal ball direction V to left-to-right (1). Draw (FOR loop) 6 colored stacked walls, each with a different paper color. Finally, display author info at the bottom of the screen by printing to stream #1.

```
1 BORDER 1:INK 0:PAPER 0:CLS:LET L=6:LET A=23214:LET V=1:FOR I=32 TO 223
2 POKE 22560+I,8*INT(I/32):NEXT I:PRINT #1;AT 1,4;"Evas10n by Marco V. 2020"
```

Outer game loop: the game executes line 3 at the beginning and every time a ball is lost (and you have still some balls left, i.e. L > 0); see GO TO at line 6.
Initialize vertical ball direction W to up (-1), initialize ball X and Y so that it lays over the middle of the paddle. Draw the paddle (FOR loop), which is 3 characters long.

```
3 LET W=-1:LET X=A-23199:LET Y=20:LET B=A-31:FOR I=0 TO 2:POKE A+I,32:NEXT I
```

Main game loop: calculate new ball X and Y (based on previous values and current ball horizontal and vertical direction. If “z” (left) or “x” (right) key is pressed (and the paddle is not in one of the bottom screen corners), move the paddle accordingly.

```
4 LET X=X+V:IF(INKEY$="x")*(A<23229)THEN POKE A+3,32:POKE A,0:LET A=A+1
5 LET Y=Y+W:IF(INKEY$="z")*(A>23200)THEN LET A=A-1:POKE A,32:POKE A+3,0
```

Then, hide the ball by POKEing its screen address B with black (0) and update the ball screen address based on its previous value and current ball horizontal and vertical direction. Then, check if the ball is at the bottom of the screen (Y=21). If so, after BEEPing a sad note on the computer speaker, the number of available balls is decremented. The GO TO statement makes the program jump back to:

- the beginning of outer game loop (line 3) if you have at least one ball left (L>0);
- the beginning of the program (line 1) if you have no balls left. This will restart the game.

```
6 POKE B,0:LET B=B+V+32*W:IF Y=21 THEN BEEP .7,-23:LET L=L-1:GO TO 2*(L>0)+1
```

Line 7 checks if there is a brick on the left (if the ball is moving from right to left) or on the right (if the ball is moving from left to right) of the ball, by PEEKing the memory address adjacent to B.
If so, the brick in that position is hit and deleted (POKE B+V, 0), the ball horizontal direction is inverted (LET V=-V) and the ball now will move towards the bottom of the screen (LET W=1).

```
7 IF PEEK(B+V) AND X>0 AND X<31 THEN POKE B+V,0:LET V=-V:LET W=1:GO TO 9
```

Line 8 draws the ball in its new position, stored in B. Then it calculates the horizontal distance between the paddle and the ball D and its absolute value E. If the ball is over the paddle and is exactly in the middle, the distance will be 0. Instead, if the ball is on one of the top corners of the paddle, the distance E will be 2. Finally, if there is a brick exactly over the ball (PEEK B-32), the brick in that position is hit and deleted (POKE B-32, 0) and the ball now will move towards the bottom of the screen (LET W=1). The ball horizontal direction will not change.

```
8 POKE B,56:LET D=B-A+31:LET E=ABS D:IF PEEK (B-32) THEN POKE (B-32), 0:LET W=1
```

Line 9 updates the ball vertical direction W to:

- -1 (up), if the ball is being hit by the paddle (absolute distance <= 2)
- Current vertical direction W, otherwise (absolute distance > 2)

Then, if the ball hits the top of the screen (Y=0), the “FREE!” message will inform you that the game has been successfully completed and the border will be painted green (4). The game will then freeze, by means of GO TO statement endlessly jumping back to current line.

```
9 LET W=-(E<3)+(E>2)*W:IF Y=0 THEN PRINT #1;AT 0,0;"FREE!":BORDER 4:GO TO 9
```

Line 10 updates the ball horizontal direction V to:

- Its current value V, if the ball does not hit one of the paddle corners (absolute distance E is different than 2);
- 1 (→), if the ball hits the paddle right corner;
- -1 (←), if the ball hits the paddle left corner.

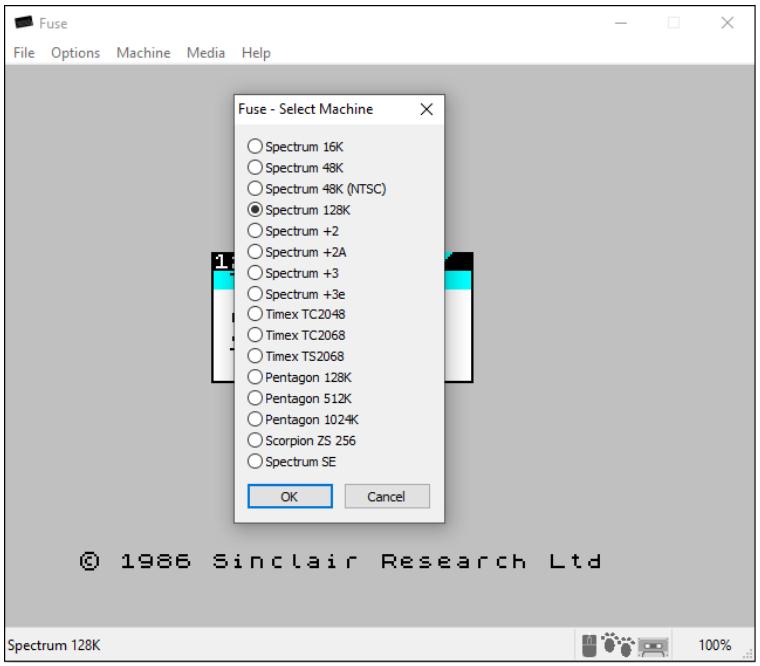
Then the updated horizontal direction is adjusted if the ball has reached either the left or the right end of the screen. Finally, the program jumps back to the beginning of the game loop, at line 4.

```
10 LET V=(E<>2)*V+(D=2)-(D=-2):LET V=(X=0)-(X=31)+V*(X<>0 AND X<>31):GO TO 4
```

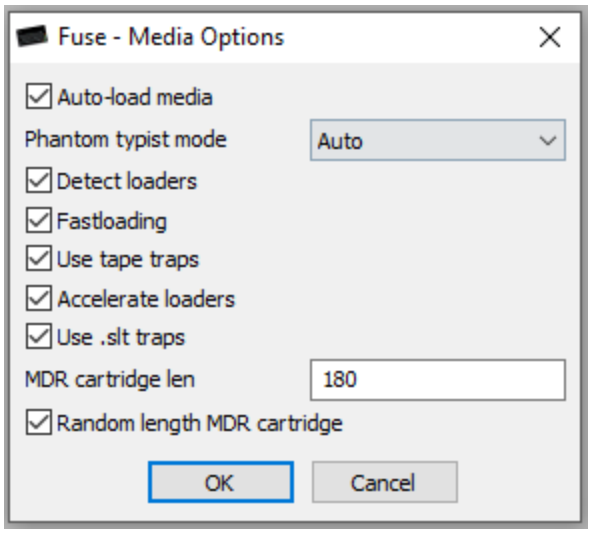
Emulator instructions

Evas10n has been tested with [Fuse emulator], which is available for many host platforms (Windows, MacOS, Linux, …). Even if Evas10n will run on both 48k and 128k ZX Spectrum configurations, if you want to access the program listing within the emulator, unless you are an experienced ZX Spectrum user, it is advisable to emulate a 128k Spectrum (see "List the program" section below).

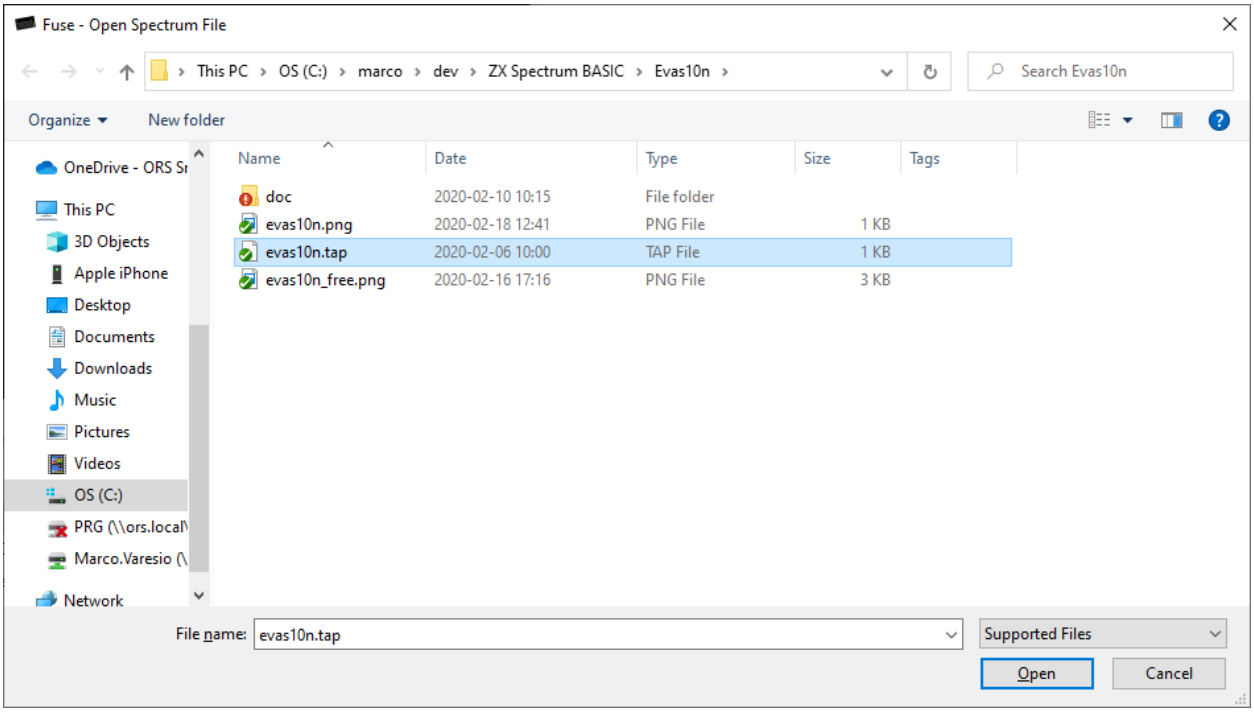
Once you have downloaded and started Fuse, you can configure the desired ZX Spectrum model from the "Machine" → "Select…" menu:



For non-experienced ZX Spectrum users, it is also advisable to enable automatic loading of tape images. This can be done by opening Media Options ("Options" → "Media") and configuring them as depicted below:



Finally, select "File" → "Open…" and then use the popup dialog to locate and open the **evas10n.tap** file.



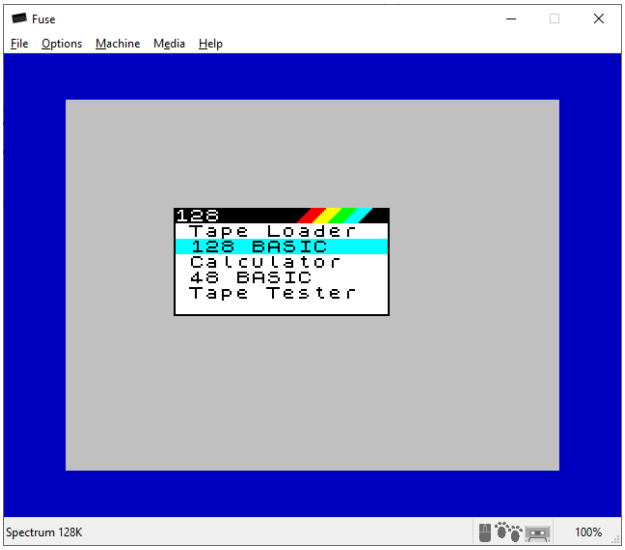
Once loaded, the game will automatically run:

You can control the green paddle with “z” and “x” keys.

Listing the program

To inspect the program source code within the emulator, it is better to have the emulated system configured as a Spectrum 128k. When successfully loaded, the program will automatically run. You can BREAK it and return to the system by pressing SHIFT + SPACE. The “L BREAK into program” message will be shown.

Then, if you press SPACE again, the Spectrum menu will be displayed.



Use cursor keys to move to “128 BASIC” and press ENTER to open the BASIC editor and the program listing will be shown.

If you run on a 48k Spectrum, after you BREAK the program using SHIFT + SPACE, you must select an INK color different than 0, and then you will be able to access the program listing by pressing “k” key and then ENTER.