Bastilude

BASIC 10-Liner action videogame for the Sinclair ZX Spectrum

Marco Varesio (Marco's Retrobits)
English language blog
Italian language blog
YouTube channel

Play online/download link
Gameplay video

Bastilude (Basic Hastilude) is an action videogame written in just 10 lines of **BASIC** programming language, for the **Sinclair ZX Spectrum** home computer.

It is my entry to the 2022 edition of the BASIC 10 Liner Contest, PUR-80 (max 80 characters per logical line, abbreviations allowed) category.

Bastilude is inspired by the Joust arcade game, developed by Williams and first released in 1982.

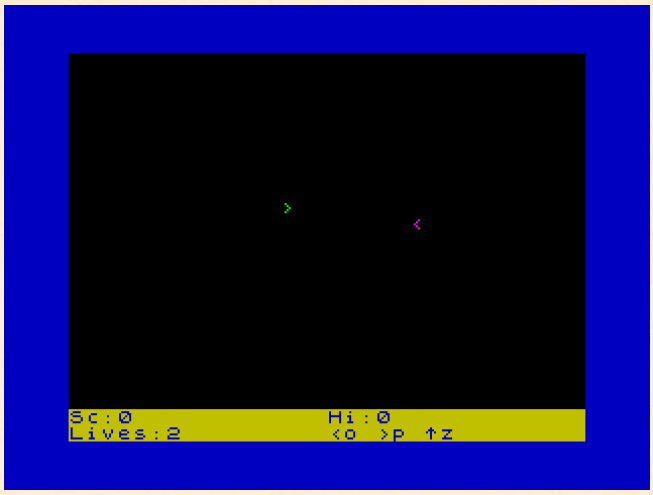
Gameplay

The player controls (with a lot of imagination) a knight riding a green dragon. The objective is to unseat the opponent knights, riding magenta dragons, in a joust, or hastilude. The winner of a joust is the rider whose mount is highest at the moment of contact. A collision of equal height repels the characters apart. The difficulty level progressively increases as the opponents are defeated and they become faster.

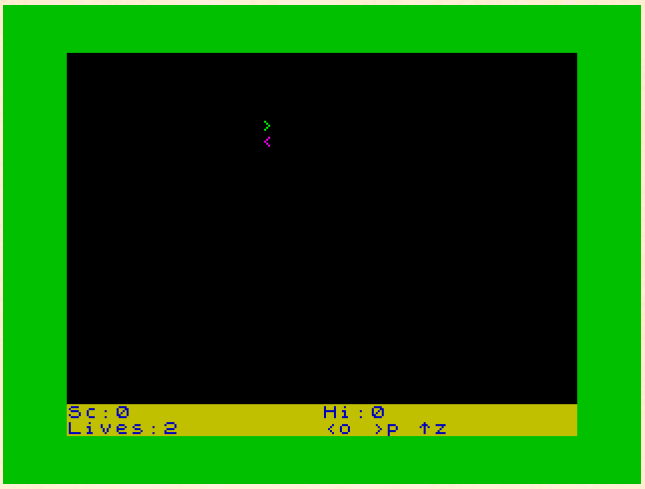
When the program starts, the title screen is shown.

Bastilude title screen

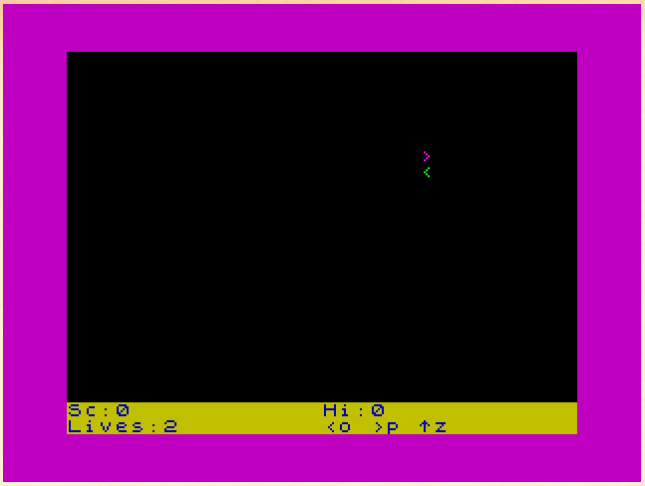
Thou can start the game by pressing any key. The 'o' and 'p' keys control the dragon's horizontal direction and velocity, while the 'z' key makes the dragon fly. Only a single command is accepted at a time, so you can either control the horizontal speed and direction or flap the dragon's wings.



Collision is imminent: prepare for jousting!

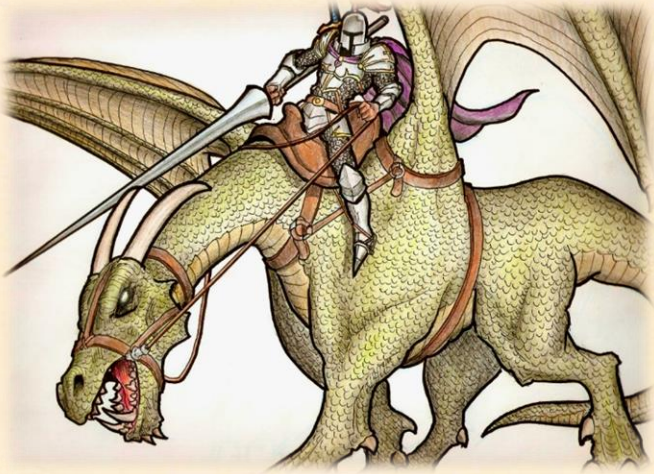


The player (green dragon) wins the joust



The opponent (magenta dragon) wins the joust

In the lower part of the screen, the program keeps track of the player's current lives left and score and of the high score. A quick reminder of the controls is also shown.



Program description

Variables

C	Player X position
R	Player Y position
X	Player X position on screen (integer part of C)
Y	Player Y position on screen (integer part of R)
V	Player horizontal velocity
W	Player vertical velocity
B	Opponent X position
A	Opponent Y position
H	Opponent X position on screen (integer part of B)
G	Opponent Y position on screen (integer part of A)
F	Opponent horizontal velocity
E	Opponent vertical velocity
L	Lives left
S	Current score
T	High score
Z	Random number 0<=Z<1
I	Loop iterator

Program listing

```
1 BRIGHT 1: BORDER 6: PAPER 0: CLS: FOR i=1 TO 7: PRINT AT i+6,i+6;INK i;"BASTILUDE":
NEXT i: LET r=21: LET c=15: IF PEEK 25102=128 THEN LET t=0

2 INK 4: LET v=0: LET w=1: LET a=10: LET b=15: LET e=.2: LET f=-.2: LET g=0: LET h=0:
LET l=2: LET s=0: GO SUB 10: PAUSE 0: BORDER 6: CLS: GO SUB 10

3 LET x=INT c: LET y=INT r: LET g=INT a: LET h=INT b: PRINT AT y,x;CHR$(61+SGN
v+(v=0));INK 3;AT g,h;CHR$(61+SGN f+(f=0)): IF h<>x THEN GO TO 6

4 IF g=y-1 THEN BORDER 3: RANDOMIZE: LET l=l-1: LET r=21: LET c=8+16*(h<15): LET v=0:
BEEP .5,-23: PRINT AT y,x;" ": GO SUB 10: GO TO 1+2*(l>-1)
```

```
5 IF y=g-1 THEN BORDER 4: RANDOMIZE: LET s=s+1: LET a=RND*22: LET b=RND*32: LET e=ABS
e+.05: LET f=ABS f+.05: BEEP .1,12: PRINT AT g,h;" ":GO SUB 10: GO TO 3

6 LET k$=INKEY$: LET v=v+.2*(k$="p")*(v<1)-.2*(k$="o")*(v>-.8): IF e>.95 THEN LET
e=.95

7 LET w=-1*(k$="z")*(y>0) +(k$<>"z")*(y<21)+(y=0): LET z=RND: IF y=g AND h=x THEN LET
f=-f: LET v=-SGN f*ABS v

8 LET c=c+v: LET c=c-32*(c>=32)+32*(c<0): LET b=b+f: LET b=b-32*(b>=32)+32*(b<0): LET
r=r+w

9 LET a=a+e*SGN(y-1-g)*(g>0)+(g=0): PRINT AT y,x;" ";AT g,h;" ": LET f=-
f*(z<.05)+f*(z>=.05): GO TO 3

10 LET t=t*(t>s)+s*(t<=s): PRINT #1;PAPER 6;INK 1;AT
0,0;"Sc:";s,"Hi:";t,"Lives:";l,"<o >p ^z": BORDER 1: RETURN
```

Source code explained

Initialization

Lines 1 and 2 display the title screen and perform variables initialization.

```
1 BRIGHT 1: BORDER 6: PAPER 0: CLS: FOR i=1 TO 7: PRINT AT i+6,i+6;INK i;"BASTILUDE":
NEXT i: LET r=21: LET c=15: IF PEEK 25102=128 THEN LET t=0

2 INK 4: LET v=0: LET w=1: LET a=10: LET b=15: LET e=.2: LET f=-.2: LET g=0: LET h=0:
LET l=2: LET s=0: GO SUB 10: PAUSE 0: BORDER 6: CLS: GO SUB 10
```

The first two lines are executed every time a new game starts, but the high score T must be defined and initialized to zero only once. This is done by inspecting the variables area in memory by means of the PEEK statement at line 1; 25102 (0x620E) is, in fact, the address at which the high score variable identifier is expected (this address has been empirically determined, based on the program length). At the first execution of the PEEK 25102 statement, only the loop variable I and the R and C variables have been defined and this location will contain the terminator value 128 (0x80), so the T variable is defined and initialized to 0. On subsequent executions, the location will contain the value 116 (0x74), corresponding to the code of the "t" character, so the comparison with 128 will fail and the T variable value will not be set to 0 anymore.

Game loop

The actual game loop starts at line 3.

```
3 LET x=INT c: LET y=INT r: LET g=INT a: LET h=INT b: PRINT AT y,x;CHR$(61+SGN v+(v=0));INK 3;AT g,h;CHR$(61+SGN f+(f=0)): IF h<>x THEN GO TO 6
```

The integer part of the player's and opponent's coordinates are computed and are used for PRINTing them on screen. Both player and opponents are depicted using the "<" (code 60) or ">" (code 62) characters, depending on their horizontal direction (facing left or right).

If the X coordinates of player and opponent differ, there cannot be any collision and the program jumps to line 6.

Otherwise, if the opponent is above the player (line 4), the player loses one life and the game status is updated. If there are no more lives left, the program jumps to line 1, otherwise the player is positioned at the bottom line of the screen and the game loop continues at line 3.

```
4 IF g=y-1 THEN BORDER 3: RANDOMIZE: LET l=l-1: LET r=21: LET c=8+16*(h<15): LET v=0: BEEP .5,-23: PRINT AT y,x;" ": GO SUB 10: GO TO 1+2*(l>-1)
```

If the player is above the opponent (line 5), the player scores one point, a new opponent appears at a random position and the opponent horizontal and vertical velocities increase. The game status is updated and the game loop continues at line 3.

```
5 IF y=g-1 THEN BORDER 4: RANDOMIZE: LET s=s+1: LET a=RND*22: LET b=RND*32: LET e=ABS e+.05: LET f=ABS f+.05: BEEP .1,12: PRINT AT g,h;" ":GO SUB 10: GO TO 3
```

Line 6 reads the keyboard status and if either the "o" (left) or "p" (right) key is pressed, the player's horizontal velocity V is updated accordingly. The opponent's vertical velocity E is limited so that it cannot become greater than 0.95, otherwise it would become impossible for the player to win the joust.

```
6 LET k$=INKEY$: LET v=v+.2*(k$="p")*(v<1)-.2*(k$="o")*(v>-.8): IF e>.95 THEN LET e=.95
```

Line 7 checks if the "z" key is pressed and updates the player's vertical velocity W accordingly. If the player is ascending and hits the top of the screen (Y=0), it "bounces" down.

The collision check is completed in the second part of line 7: if the player and opponent occupy the same position, the joust is a draw and the characters horizontal velocities are updated so that they will move to opposite directions.

```
7 LET w=-1*(k$="z")*(y>0) +(k$<>"z")*(y<21)+(y=0): LET z=RND: IF y=g AND h=x THEN LET
f=-f: LET v=-SGN f*ABS v
```

Line 8 updates both player's and opponent's X positions (C and B) based on horizontal velocities V and F. The player's Y position R is updated based on vertical velocity W.

```
8 LET c=c+v: LET c=c-32*(c>=32)+32*(c<0): LET b=b+f: LET b=b-32*(b>=32)+32*(b<0): LET
r=r+w
```

In line 9, the opponent's Y coordinate A is updated based on vertical velocity E and player's Y position; in fact, the opponent tries to fly one position above the player in order to win the joust. Then, both characters are erased from screen; they will be displayed at the updated coordinates in the next game loop iteration. To add some unpredictability to the game, with a probability of 5% the opponent's horizontal direction is inverted. Finally, the program jumps back to the beginning of the game loop at line 3.

```
9 LET a=a+e*SGN(y-1-g)*(g>0)+(g=0): PRINT AT y,x;" ";AT g,h;" ": LET f=-
f*(z<.05)+f*(z>=.05): GO TO 3
```

Print game status routine

The subroutine at line 10 first checks if current score S is greater than high score T and updates T accordingly. Then, it prints current score, high score, player's lives left and a reminder of the game controls at the bottom of the screen. Before returning to the caller, the border colour is restored to blue.

```
10 LET t=t*(t>s)+s*(t<=s): PRINT #1;PAPER 6;INK 1;AT
0,0;"Sc:";s,"Hi:";t,"Lives:";l,"<o >p ^z": BORDER 1: RETURN
```

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:

```
1B1:b6:C0:v:fi=1 F7:pIi+6,i+6;Xi;"BASTILUDE":ni:lr=21:lc=15:uO25102=128 Glt=0
2X4:lv=0:lw=1:la=10:lb=15:le=.2:lf=-.2:lg=0:lh=0:ll=2:ls=0:h10:m0:b6:v:h10
3lx=Rc:ly=Rr:lg=Ra:lh=Rb:pIy,x;U(61+Fv+(v=0));X3;Ig,h;U(61+Ff+(f=0)):uh<>x Gg6
4ug=y-1 Gb3:t:ll=l-1:lr=21:lc=8+16*(h<15):lv=0:Z.5,-23:pIy,x;" ":h10:g1+2*(l>-1)
5uy=g-1 Gb4:t:ls=s+1:la=T*22:lb=T*32:le=Ge+.05:lf=Gf+.05:Z.1,12:pIg,h;" ":h10:g3
6lk$=XEY$:lv=v+.2*(k$="p")*(v<1)-.2*(k$="o")*(v>-.8):ue>.95 Gle=.95
7lw=-1*(k$="z")*(y>0) +(k$<>"z")*(y<21)+(y=0):lz=T:uy=g Yh=x Glf=-f:lv=-Ff*Gv
8lc=c+v:lc=c-32*(c>=32)+32*(c<0):lb=b+f:lb=b-32*(b>=32)+32*(b<0):lr=r+w
9la=a+e*F(y-1-g)*(g>0)+(g=0):pIy,x;" ";Ig,h;" ":lf=-f*(z<.05)+f*(z>=.05):g3
10lt=t*(t>s)+s*(t<=s):p#1;C6;X1;I0,0;"Sc:";s,"Hi:";t,"Lives:";l,"<o >p ^z":b1:y
```

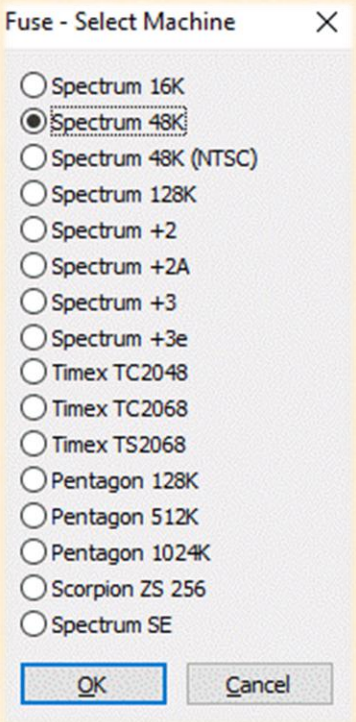
Therefore, **Bastilude** is a suitable entry to the PUR-80 category of the BASIC 10 Liner Contest.

Loading instructions

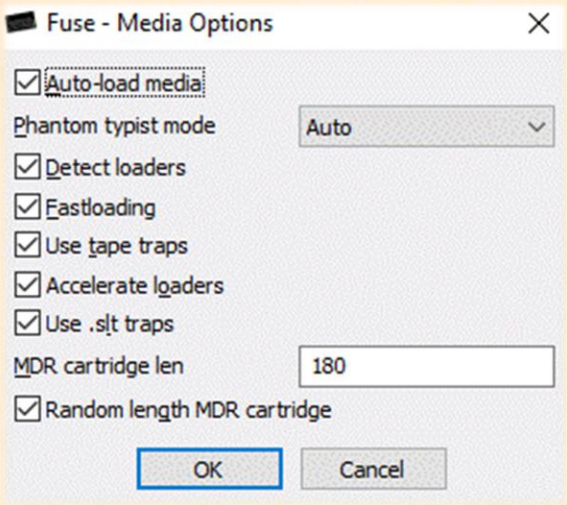
Bastilude is provided in TAP tape image format, which can be easily loaded in most ZX Spectrum emulators and on the real machines, either equipped with devices such as the DivMMC or by playing it through the MIC port using tools like PlayTZX or WinTZX.

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.

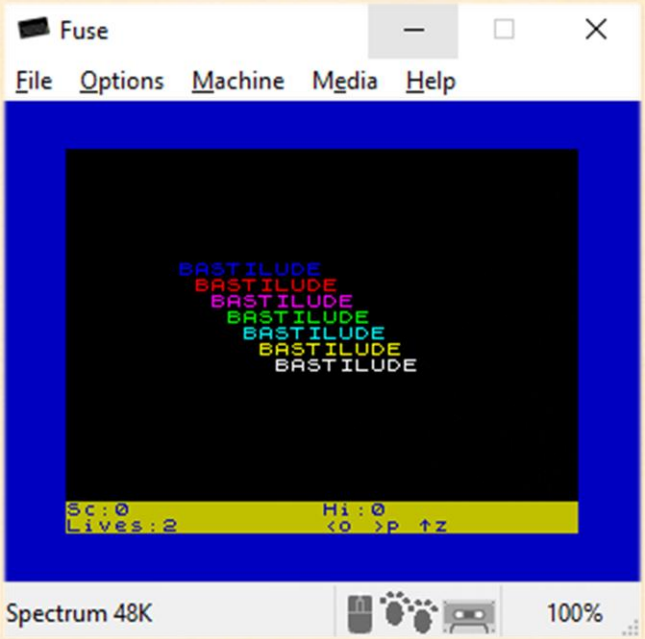
Start the Fuse emulator and select the Spectrum 48K model in "Machine" -> "Select..."



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

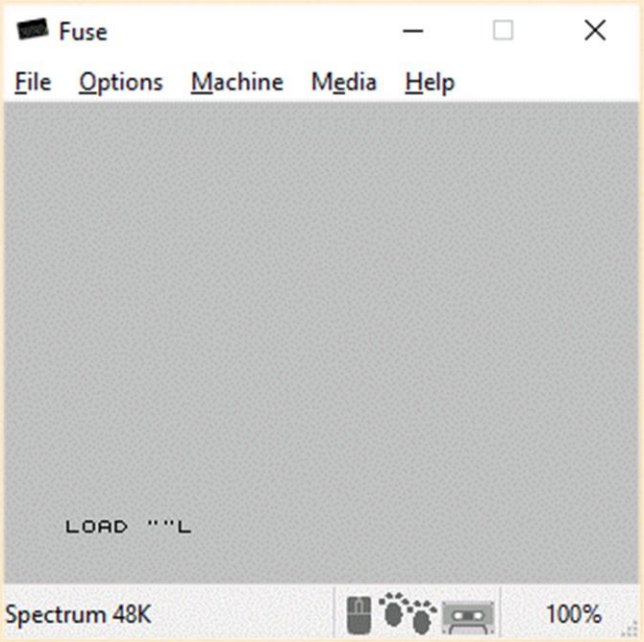


Open the **bastilude.tap** file, either by selecting it in "File"->"Open..." or by dragging and dropping it on the emulator window.



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.

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.



References

Joust Wikipedia entry

Joust Atari 2600 manual

Images used in this document

This document has been enriched with the following pictures. Upon request from the authors or copyright holders, they will be removed from the document.

Parchment scroll background public domain from wpclipart.com

Flying dragon picture from 4ever.eu

Girocomic official poster by Rafael Teruel (rafater)

Dragon rider by Jazza

A knight riding a dragon from pics.alphacoders.com

Knight riding on flying dragon by Martin Davey