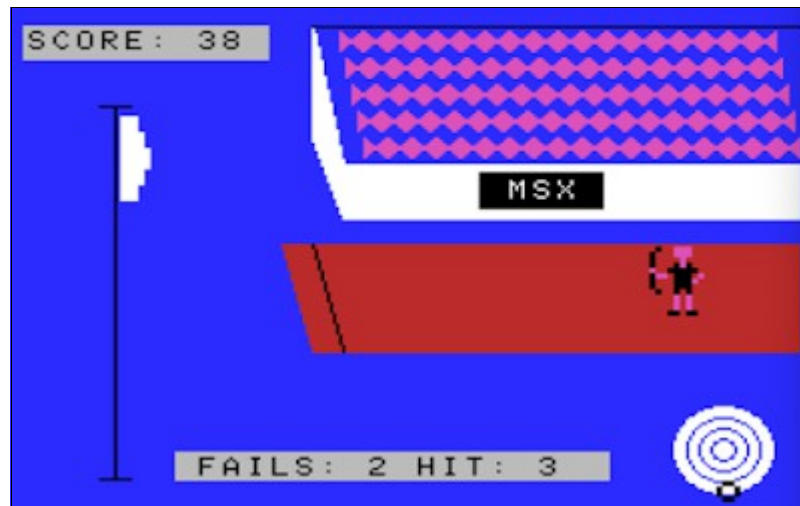


Archer10. Submission to the 2021 10-Liner Contest

Some weeks ago I had notice of a retro programming contest, yes through one of these wasap groups, the [10th Edition of BASIC 10 Liner Contest](#). The main idea is that you develop a game in BASIC with no more than 10 lines of code for 8 bit legacy computers. There are different categories depending on the maximum number of characters per line allowed. Simple, isn't it? I decided to go for a try. I though "come on, it is only 10 lines of code, it can not be of much pain". I could not have been more wrong. In this post I will explain what and how I developed my submission for this year edition (2021), which I have named **Archer10**.



The gameplay

Being the machine I learned with I decided to go for a MSX-1 program. Let's call it nostalgia. Well, and it is the machine that I actually know more of. Once the machine is settled, next thing is thinking on a gameplay. I am not good at all with games, so I decided to perform some plagiarism of myself. I started with some ideas borrowed from [Barcelona 92](#), a BASIC program I developed when I was 15 years old and was published in the magazine *MSX Club de Programas* nº 3-4 (1985).

After some not-too-deep-thinking I came up with a very simple gameplay: a target on the left moves up and down. You control an archer on the right who throws arrows from right to left. If you hit the target you get points depending on how close to the center the arrow impacts. You are allowed a maximum number of fails. Then the program finishes. The larger the final score, the better. Well, quite simple, but definitively a game. And it has some nice features I can exploit. Something I dislike of many games is that they are very hard from the beginning. Being very bad at gaming, this disallows me from trying them. So I decided the game at the beginning was to be very easy (the target moves very slowly). But then, the difficulty gets increasing little by little (you hardly notice the difference at first) so that after some throws it gets really difficult. And the constraint in the number of fails makes you to try again to increase your score.

Good. I had a gameplay. Now the visual part. I wanted to expose some of the very-nice-features of the MSX-1 machines, so I decided to include high resolution graphics (known as SCREEN 2 in MSX jargon), hardware sprites, and sound. With plenty of colors, of course (disclaimer: I am not color impaired, but definitively I am not good at coloring). And weird sounds as well (disclaimer: I am as good at music as I am at coloring).

The programming

I started coding a mock up of the program without constraints, with many lines, to get it up and running. I wanted to make sure the gameplay was as I intended. Then I worked the graphics so to avoid the color attribute clash (typical of the MSX-1 video processor), but at the same time displaying most of the available color palette.

I roughly had over a hundred lines of code with some 3 Kb of program space. Definitely too much for a 10-liner.

Because of the graphics I was sure I needed quite a lot program space. In addition having text in a graphics screen is quite a lot program space consuming. And I definitely wanted some text. No compromises allowed.

Thus I decided to enter the EXTREME-256 category. In addition to only ten lines of code, each line must not contain more than 256 characters. Overall, the code can not be larger than 2.5 Kb !!! To squeeze the code in this size you play the very old tricks. The BASIC interpreter uses a lexical analyzer that is based on keyword detection and not in delimiters. This basically means that you can write the code without white spaces. Who cares about readability !!! Then you use one-letter variables and single-digit line numbers (after all, remember, you can not use more than 10 :-).

Now you get a very compact BASIC code with the minimum of program storage needed. This is the simple stuff. But there is more than this, you have to actually code the gameplay, which involves actions and decisions. When programming we tend to think on logical sequences (at least, we should do that) which end up coded using control structures with GOTOs and, for the sake of clarity, with GOSUBs. The problem here is that with the reduced number of lines available GOTOs and GOSUBs are of very limited use. So instead thinking on logical sequences, where you have actions and decisions clearly isolated, you have to interleave all of them, fitting as much as possible in a single line of code. It is a very interesting programming exercise that you do not typically practice.

The code

If I say I finished up with 10 lines of code, this should not be of great surprise. No spoiler here. Lines 0 to 3 initialize the program and do all the text and graphics background drawing and hardware sprites definition. The order in which instructions are performed are related to the available line space. I tried different combinations. One interesting thing is that only line 3 is needed to replay, so you can GOTO 3 when the game is finished to start a new one.

Lines 4 to 7 implement the gameplay. They manage arrow and target movement, user input and scoring. I was lucky enough to implement a single subroutine in line 7, with is called as GOSUB 7 each time the arrow reaches the left part of the screen (being it a hit or a fail). Line 6 plays a very nice trick. It checks for both the end of the game and the reinitialization of it (after pressing the space bar). The later issuing a GOTO 6. This corrupts the state of the game, but as it has finished, who cares!!!. When you do GOTO 3 the game state is correctly initialized.

Lines 8 to 9 contain DATA definitions for the hardware sprites coded in hexadecimal. I must say this is NOT the most efficient way of storing this information. You can use string literals, but I sacrificed some program storage (with I finally did not need) for character coding. These literals might (and do) contain characters which are not ASCII standard. As I develop both on my Mac and on my Sony HB-F700S it produced me many headaches while copying to and from and trying to preserve character coding.

The state of the game is coded using the following variables:

- **Y** (Float): vertical position of the target in pixels. Varies in the 30 .. 135 range
- **S** (Float): amount in pixels which is added/subtracted to Y at each time step. It controls the velocity of the target. Its initial value is 1.0. Each time a hit occurs it is increased by 0.25
- **X** (Integer): horizontal position of the arrow in pixels. Varies in the 200 .. 22 range

- **G** (Integer): total score achieved
- **F** (Integer): number of fails
- **H** (Integer): score of the last hit. Computed as a linear distance between the arrow y position and the target y position
- **B** (Integer): state variable representing whether the arrow is moving (B=1) or not (B=0)

I finally had a code of 2.1 Kb, in 10 lines of BASIC, with standard ASCII character coding. Nice. Goal achieved. You can find the code, a simulator disk file (DSK) and some multimedia materials here:

GitHub: <https://github.com/humbertomb/mymx/tree/master/archer10>

Instructions

You are a top-notch archer in the "*Outdoor Archery World Series. Moving Target*". You throw an arrow by pressing the space bar, and cannot throw a new one until it either makes a hit or a fail. The target moves so you have to carefully synchronize the arrow throwing with the target movement. The target speed gets increasing with each hit. You get points for each hit depending of how close the arrow gets to the target center. You are allowed a maximum of 3 failures, in which case the competition finishes and you get your final score. To start a new game hold the space bar for more than one second.



Are you ready? Can you make more than one hundred points? Let's try it online.

Download the file [archer10.dsk](#) from GitHub and drag&drop it on the blue console of [WebMSX](#). A pop up window appears and select *Dive A*. Then the simulator loads the disk and it is ready to accept commands. Type the following command in the console and then press INTRO:

```
RUN "ARCHER10.BAS"
```

Too complex?

I have prepared a web page which automatically loads and executes the program in WebMSX. [Follow this link to play online](#)

The game has sound, so remember to enable audio in your computer.

Enjoy playing!!