MSX Article

MARMSX

Fade in & Fade out

Summary

This article aims at presenting a well-known graphic effect called fade, which consists in presenting or removing images through brightness variation.

1- Fade – The Brightness Control

The fade effect is widely used on movies and games, usually when a new scene comes up or finishes. It is the transition from black background to the image or vice-versa by means of brightness variation.

MSX 2 palette allows us to control the color brightness in an easy and fast way, once each palette entry controls all pixels labeled with that index together. In this article, MSX 1 color set will be used, although any other may be used.

The variable F will control image brightness, ranging from 0 to 7. But, the question is how to control different palette settings using only one variable?

Once the highest MSX 2 palette value is 7, when F reaches this value, all original palette values must be recovered. In other words, when F is 7, the image has it original brightness. In that case, it is easy to deduct the following formula that controls the brightness for each color component in each palette entry:

$$C_t = \frac{F \times C}{7}$$

Where C_t is the transitory color, and C the original color. Notice that when F=7, C_t=C.

We will present now two different solutions in Basic to perform fade-in (presentation) and fade-out (removing) animations on a screen 7 image.

The steps for both solutions are:

1. Set all palette colors to 0, in order to make the image invisible as soon it is loaded.
2. Load the image
3. Start fade-in process
4. Wait for the user press “enter”
5. Start fade-out process
6. Finish the program

Fade1.bas

```
10 SCREEN 7
20 FOR I=0 TO 15
30 COLOR=(I,0,0,0)
40 NEXT I
50 BLOAD"IMAGE.S07",S
60 FOR F=0 TO 7 STEP 1                  Fade in
70 GOSUB 500
80 NEXT F
90 A$=INPUT$(1)
100 FOR F=7 TO 0 STEP -1                Fade out
110 GOSUB 500
120 NEXT F
130 SCREEN 0
140 COLOR = NEW
150 END
500 COLOR=(2, F*1/7, F*6/7, F*1/7)
510 COLOR=(3, F*3/7, F*7/7, F*3/7)
520 COLOR=(4, F*1/7, F*1/7, F*7/7)
530 COLOR=(5, F*2/7, F*3/7, F*7/7)
540 COLOR=(6, F*5/7, F*1/7, F*1/7)
550 COLOR=(7, F*2/7, F*6/7, F*7/7)
560 COLOR=(8, F*7/7, F*1/7, F*1/7)                 Subroutine for
570 COLOR=(9, F*7/7, F*3/7, F*3/7)                 updating palette
580 COLOR=(10, F*6/7, F*6/7, F*1/7)                values
590 COLOR=(11, F*6/7, F*6/7, F*4/7)
600 COLOR=(12, F*1/7, F*4/7, F*1/7)
610 COLOR=(13, F*6/7, F*2/7, F*5/7)
620 COLOR=(14, F*5/7, F*5/7, F*5/7)
630 COLOR=(15, F*7/7, F*7/7, F*7/7)
640 RETURN
```

The MSX 8-bit processor is quite slow for calculations. In addition, the Basic language is interpreted, resulting on very slower calculations. In that case, multiplications and divisions inside a loop will result in a quite slow animation. The previous algorithm took about 5 seconds to complete animation from black background to the image.

In order to improve the performance for that algorithm, the palette intermediary values corresponding to F will be pre-calculated and stored in a matrix. So, when the animation starts, the program will only read a value instead of calculating it.

The steps for this new solution are:

1. Set all palette colors to 0, in order to make the image invisible as soon it is loaded.
2. Pre-calculate palette values
3. Load the image
4. Start fade-in process
5. Wait for the user press “enter”
6. Start fade-out process
7. Finish the program

Fade2.bas

```
10 SCREEN 7
20 FOR I=0 TO 15
30 COLOR=(I,0,0,0)
40 NEXT I
50 DIM P(7,7)
60 GOSUB 400
70 BLOAD"IMAGE.S07",S
80 FOR F=0 TO 7 STEP 1
90 GOSUB 500
100 NEXT F
110 A$=INPUT$(1)
120 FOR F=7 TO 0 STEP -1
130 GOSUB 500
140 NEXT F
150 SCREEN 0
160 COLOR = NEW
170 END
400 FOR F=1 TO 7
410 FOR C=1 TO 7
420 P(F,C) = F * C/7
430 NEXT C, F
440 RETURN
500 COLOR=(2, P(F,1), P(F,6), P(F,1))
510 COLOR=(3, P(F,3), P(F,7), P(F,3))
520 COLOR=(4, P(F,1), P(F,1), P(F,7))
530 COLOR=(5, P(F,2), P(F,3), P(F,7))
540 COLOR=(6, P(F,5), P(F,1), P(F,1))
550 COLOR=(7, P(F,2), P(F,6), P(F,7))
560 COLOR=(8, P(F,7), P(F,1), P(F,1))
570 COLOR=(9, P(F,7), P(F,3), P(F,3))
580 COLOR=(10, P(F,6), P(F,6), P(F,1))
590 COLOR=(11, P(F,6), P(F,6), P(F,4))
600 COLOR=(12, P(F,1), P(F,4), P(F,1))
610 COLOR=(13, P(F,6), P(F,2), P(F,5))
620 COLOR=(14, P(F,5), P(F,5), P(F,5))
630 COLOR=(15, P(F,7), P(F,7), P(F,7))
640 RETURN
```

This new solution took only 2 seconds to present the image, 3 seconds less than the previous solution. It is clear that if this solution were coded in Assembly, the performance would increase considerably.

2- Credits

This article was written by Marcelo Silveira, October 2016.

E-mail: flamar98@hotmail.com
Homepage: http://marmsx.msxall.com