MSX Article

Fade in e Fade out

Resumo

O objetivo deste artigo é apresentar um recurso gráfico comum chamado de fade, que consiste em apresentar (ou retirar) uma tela através da variação de brilho, desde o mais fraco até o mais forte.

1- Fade - Controlando o brilho

O recurso de fade é comum em filmes e jogos eletrônicos, e consiste em apresentar ou retirar imagens da tela através da variação de brilho.

O brilho de uma tela no MSX 2 pode ser controlado através da manipulação da paleta de cores, presente nas screens de 0 a 7.

A paleta utilizada aqui será a paleta de cores padrão do MSX 2. A variação de brilho será controlada pela variável F, que varia de 0 a 7.

A questão é como introduzir a variável F de modo a controlar o brilho da paleta?

Sabe-se que o valor máximo de brilho no MSX 2 é 7. Assim, quando esse valor for atingido, deverá restaurar os valores originais da paleta utilizada.

Por exemplo, se o canal de cor vermelho do índice 2 da paleta tiver intensidade igual a 1, então, quando F for igual a 7, o valor dessa componente deverá ser igual 1.

Chamando-se de “C” o valor de intensidade de cor original de cada canal, tem-se que:

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

Onde “Cor” é a cor transitória de cada componente de cor da paleta de cores.

Dessa forma, quando F valer 7, a variável “Cor” valerá a cor original “C”.

A seguir, serão apresentadas duas soluções em Basic para fazer o fade in (apresentação) e o fade out (retirada) de uma imagem na screen 7.

Os principais passos dessas soluções são:

1. Atribuir a todas as cores da paleta o valor igual a 0, para que a imagem fique “invisível” ao ser carregada.
2. Carregar a imagem
3. Iniciar o processo de fade in
4. Aguardar o usuário teclar “enter”
5. Iniciar o processo de fade out
6. Terminar o programa

Fade1.bas

```
10 SCREEN 7
20 FOR I=0 TO 15
30 COLOR=(I,0,0,0)
40 NEXT I
50 BLOAD"IMAGEM.S07",S
60 FOR F=0 TO 7 STEP 1
70 GOSUB 500
80 NEXT F
90 A$=INPUT$(1)
100 FOR F=7 TO 0 STEP -1
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)
570 COLOR=(9, F*7/7, F*3/7, F*3/7)
580 COLOR=(10, F*6/7, F*6/7, F*1/7)
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
```

Fade in

Fade out

Sub-rotina de atualização da paleta de cores.

O MSX possui um processador de 8-bits lento para cálculos, se comparado com os processadores atuais. Assim, ao ser realizada várias iterações (repetições) com cálculos de multiplicação e divisão, a animação torna-se lenta. O algorítimo anterior levou cerca de 5 segundos para apresentar a tela, desde o brilho mais escuro até o mais claro.

De forma melhorar o desempenho do algorítimo, os valores intermediários da paleta agora serão pré-calculados, antes da animação entrar em execução. Assim, o comando color somente consulta um valor, em vez de calculá-lo.

Os principais passos dessa nova solução são:

1. Atribuir a todas as cores da paleta o valor igual a 0, para que a imagem fique “invisível” ao ser carregada.
2. Pré-calcular os valores da paleta
3. Carregar a imagem
4. Iniciar o processo de fade in
5. Aguardar o usuário teclar “enter”
6. Iniciar o processo de fade out
7. Terminar o programa

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"IMAGEM.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
```

Este algorítimo conseguiu apresentar a tela em 2 segundos, levando 3 segundos a menos que o algorítimo anterior.

É evidente, que se esse último algorítimo fosse convertido para assembly, o desempenho seria ainda muito melhor.

2- Créditos

Este artigo foi escrito por Marcelo Silveira, em Outubro de 2016.
E-mail: flamar98@hotmail.com