MSX Article

MARMSX

RGB to Gray

Summary

This article demonstrates how to convert a color image to grayscale using MSX 2.

1- Introduction

A digital image is composed by a combination of 3 primary additive colors: red, green and blue. This color system is called RGB. Do not confuse this system with the subtractive color system, which is the one used on drawings and printings. The first one is based on light emission, while the second is based on light reflection. Figure 1 shows both systems.

a) Additive color system   b) Subtractive color system

Figure 1. Color systems

Each color on additive system is composed by a mix of primary colors, each one composed by a discrete shade from the dark to bright. The number of shades or levels depends on the video processor hardware. The PCs have 256 levels on each color channel, reaching 16 million colors. The MSX 2 has 8 levels on each color channel that together represents up to 512 different colors.

In a grayscale image, each pixel have the same intensity on all color channels. In that case, we may represent a pixel using only one component – the grayscale.

We may achieve the grayscale color by calculating the arithmetic mean of red, green and blue colors and applying the result on all channels:

```
GRAY = (RED + GREEN + BLUE) / 3
```

Thus, there is a formula used in image processing that is based on the human eye response for each color component. This is a weighted mean.

```
GRAY = RED*0,3 + GREEN*0,59 + BLUE*0,11
```

Is it perfectly possible to convert from a color image to grayscale, once the grayscale is a subset of the color space. Nevertheless, the reverse way is complex, once we are trying to reach the whole set starting from a subset.

2- Conversion to Grayscale on MSX

2.1- Screens 2-7

MSX screens from 2 to 7 have theirs colors controlled by a palette system. In other words, each pixel holds an index to a table that contains the RGB color value. In that case, each palette entry controls many pixels together.

In order to convert an image from RGB to gray, we have only to change the palette color data. Quite simple, once we have to change only 16 values against 27 k or 54 k values if pixel color were controlled directly.

MSX 2 has the Basic command “COLOR=” that allows us to change the palette data. Nevertheless, there is no command that is capable of reading the palette data. In that case, we must read that data directly from the VRAM.

The VRAM area that holds the palette data changes according to the screen. The table 1 describes the VRAM area for each screen mode.

Screen	Initial address	Final address
0 / width 40	&H0400	&H041F
0 / width 80	&H0F00	&H0F1F
1	&H2020	&H203F
2	&H1B80	&H1B9F
3	&H2020	&H203F
4	&H1B80	&H1B9F
5	&H7680	&H769F
6	&H7680	&H769F
7	&HFA80	&HFA9F

Table 1. Palette address in VRAM.

Each palette index is stored in the VRAM, using 2 bytes configured as follows:

```
E   - 0rrr0bbb
E+1 - 00000ggg
```

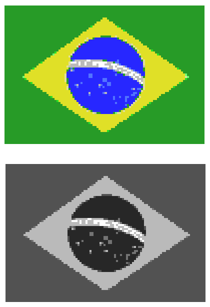
The “r” represents the red channel bits, “g” the green bits and “b” the blue bits.

Each palette entry address is calculated as follows:

```
E = initial_address + index x 2
```

The following Basic program converts any palette on screen 5 to grayscale. Draw something or load an image to see the results.

```
10 SCREEN 5
20 FOR E=&H7680 TO &H769F STEP 2
30 R = FIX(VPEEK(E)/16)
40 G = VPEEK(E+1)
50 B = VPEEK(E) AND 7
60 C = FIX(R*0.3 + G*0.59 + B*0.11)
70 VPOKE E, C*16 + C
80 VPOKE E+1,C
90 NEXT E
100 COLOR=RESTORE
110 GOTO 110
```



For the other screens, change the VRAM area address (line 20).

2.2- Screen 8

Screen 8 represents each pixel using a RGB value directly. Once MSX 2 has 3 bits to represent a intensity from each color channel ($2^3 = 8$ levels), it would be necessary 9 bits to store a color. While a byte can only store 8 bits, the MSX designers decided to remove one bit from the blue channel, once this color is the least perceived by the human visual system.
Each screen 8 pixel has the following configuration:

Bit	7	6	5	4	3	2	1	0
Color	G	G	G	R	R	R	B	B

Two algorithms will be presented to convert a color image to grayscale in screen 8. The first one is in Basic, while the second is in Assembly. The Basic program takes a few minutes to complete the operation, while the Assembly program takes around 1 minute to convert the image. The basic idea of both programs is to read each pixel by separating the color components, converting the blue channel from 2 bits to 3 bits, calculating the average color (gray) and finally setting the gray color to each channel. The arithmetic average is used in order to simplify the calculations.

```
10 SCREEN 8
20 BLOAD"image.pic",S
30 FOR Y=0 TO 211
40 FOR X=0 TO 255
50 C=POINT(X,Y)
60 B=(C AND &B00000011)*2
70 R=(C AND &B00011100)/4
80 G=(C AND &B11100000)/32
90 C=INT((R+G+B)/3)
100 B=C/2
110 R=C*4
120 G=C*32
130 C=R+G+B
140 PSET(X,Y),C
150 NEXT X,Y
160 GOTO 160
```

The Assembly code equivalent to the previous program:

```
ORG &HC000
LD D,&HD4          ; End address of screen 8
LD E,0             ;
LD HL,0            ; Initial address of screen 8
s1: LD IX,&H10D    ; RDVRM (read VRAM) Color stored in A
CALL &H15F         ; Call subrom
PUSH DE            ; Save DE
LD D,0             ; Clear mean variable D
LD E,A             ; Save the color in E
AND &B00000011     ; Get the blue channel
SLA A              ; Convert from 2 bits to 3 bits (because of R and G)
LD D,A             ; Store blue color in D
LD A,E             ; Get the full color
AND &B00011100     ; Get the red channel
SRL A              ; Shift from 000RRR00 to
SRL A              ; 00000RRR
ADD A,D            ; Add red and blue
LD D,A             ; Save the result in D
LD A,E             ; Take full color again
AND &B11100000     ; Get the green channel
LD B,5             ;
e1: SRL A          ; Shift GGG00000 to 00000GGG
DJNZ e1            ;
ADD A,D            ; Add green to sum stored at D
LD D,FF            ; Clear D (now used to store division)
LD B,3             ; Average by 3
e2:SUB B           ; A = A - B
INC D              ; D = D + 1
JR NC,e2           ; While not negative, loop e2
LD A,D             ;
SLA A              ; Do 00YYY000
SLA A              ;
SLA A              ;
ADD A,D            ; Do 00YYYYYY
SLA A              ;
SLA A              ; Do YYYYYY00
SRL D              ; Convert gray value to 2 bits
ADD A,D            ; Finallly do YYYYYYYY
LD IX,&H109        ; WRTVRM (write)
CALL &H15F         ; Call subrom
INC HL             ; Next pixel
POP DE             ; Check DE
LD A,D             ;
CP H               ;
JR NZ,s1           ; Check if reached the end of memory
LD A,E             ;
CP L               ;
JR NZ,s1           ;
RET                ; Return
```

3- Credits

This article was written by Marcelo Teixeira Silveira, originally for the MSX Rio 2008 meeting fanzine.

Date: March 2008.
Revision: July 2017.
E-mail: flamar98@hotmail.com
Homepage: marmsx.mxsall.com

Note: this is a translation from the original article titled “RGB to Gray”, in portuguese, written by the same author.