GD2 COLOUR TUTORIAL

At a QL show I once made a proud boast that in half an hour I could teach almost any one to program in the new colours using just eleven lines of SuperBasic and no pointer environment jargon.

I heard a trader mutter under his breath, "They will be very long lines!"

Well, you can judge for yourself because I am about to repeat the promise. Almost anyone who works through this tutorial can learn how to program in the new colours using a SuperBasic program of just eleven lines. And there will be no pointer environment jargon. There is no point in reading this article unless your machine is fired up and ready to go. Of course it must be able to handle the new colours. That means you have QPC2, a Q40/60, a QXL or an Aurora card using a version of SMSQ/E above 3.00.

But first a little bit of theory.

The new colours first became available in 1999 for the Q40, and a year later for QPC and the QXL. There was a much longer wait before they were introduced on the Aurora Card. For various technical reasons each QL platform had its own implementation of the new colours. For example, the Aurora card has only limited memory available and can only handle 256 colours compared with thousands for the other platforms.

Because each platform had its own colour implementation, in the early days you had to put a command in any SuperBasic program to tell it which colours to use. There were four commands:

COLOUR_QL were the standard QL colours. However you could modify these using the PALETTE_QL.

COLOUR_PAL was a palette of 256 colours. Again these could be modified by the user via the command PALETTE_QL

COLOUR_24 were true colours or 24 bit colours. However no present QL platform can handle 24 bit colours. You were also told to enter values in HEX.

COLOUR_NATIVE were the colours that your QL platform used.

If you are finding this a little complicated you are in good company. Quite a few program developers and traders also found it complicated. In particular it was difficult for them to write programs that would look the same on all platforms. It was no coincidence that in the first few years we had the new colours scarcely any software was written using them.

What the QL urgently needed was a much simpler way of using the new colours that was common to all platforms. That happened in 2003 when Marcel Kilgus developed a new Window Manager able to display the new colours.

"Window Manager?" Isn't that something to do with the dreaded Pointer Environment?

Yes it is, and if you use the new colours you will be using part of the pointer environment. But don't panic. The pointer environment is already built into SMSQ/E and the good news is that you don't need to know anything about it to use the new colours. You can use them in a simple SuperBasic program, which is what we are now going to do.

So back to our machines. First of all we are going to be lazy and use the Toolkit 2 command to set our windows, and then we are going to use white ink on a black background:

```
10 WTV
20 INK 7 : PAPER 0 : CLS
```

Now we are going to modify Window #2 to give it a black background and a white border:

```
30 WINDOW #2, 300,154,50,50
40 PAPER #2, 0 : CLS #2
50 BORDER #2,2,7
```

Now an input command:

```
60 AT 2,3 : INPUT "Colour: ";colour
```

Now a block of colour:

```
70 BLOCK #2, 292,50,0,0,colour
```

We repeat this line twice with some slight modifications:

```
80 BLOCK #2, 292,50,0,50,colour+5
90 BLOCK #2, 292,50,0,100,colour+10
```

Now add a pause command:

```
100 PAUSE
```

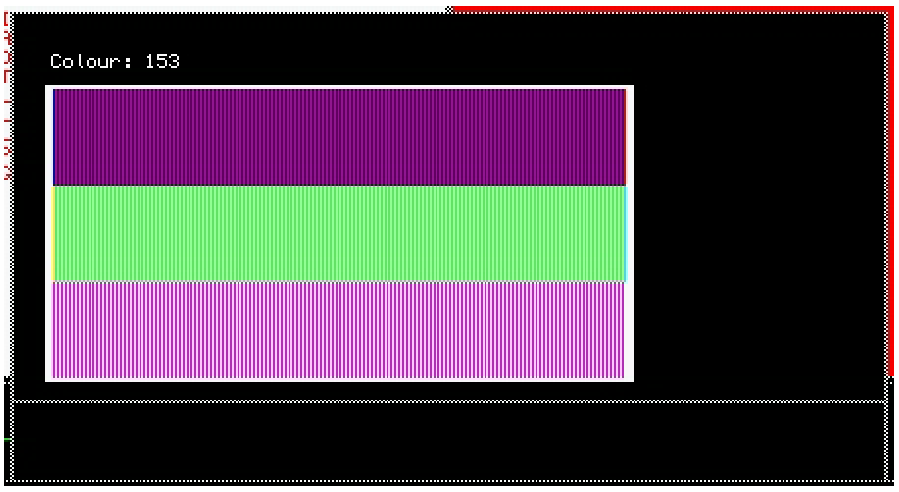
And now for something really naughty:

```
110 GO TO 60
```

Let's just print that again in one piece.

```
10 WTV
20 INK 7 : PAPER 0 : CLS
30 WINDOW #2, 300,154,50,50
40 PAPER #2, 0 : CLS #2
50 BORDER #2,2,7
60 AT 2,3 : INPUT "Colour: ";colour
70 BLOCK #2, 292,50,0,0,colour
80 BLOCK #2, 292,50,0,50,colour+5
90 BLOCK #2, 292,50,0,100, colour+10
100 PAUSE
110 GOTO 60
```

Now I count eleven lines, and not a long line in sight! Think of a number between 1 and 245 and make a note of it. Run the program, input that number and you should have something like this:


```
Colour: 153
```

Now we are going to modify a part of this program to use the new colours. Break into the program (CTRL + SPACE) and put WM_ before each BLOCK keyword in lines 70 - 90. These lines should now appear as:

```
70 WM_BLOCK #2, 292,50,0,0,colour
80 WM_BLOCK #2, 292,50,0,50,colour+5
90 WM_BLOCK #2, 292,50,0,100,colour+10
```

WM means "Window Manager" and the new keyword instructs the QL to use the new colours. Run the program inputting the number you used before and Hey Presto' it looks just the same as it did using the old colours! This is because it is meant to look the same.

The Window Manager colours are not a single series of colours, but several groups of colours. **The first group of 256 colours is exactly the same as the QL colours.**

Now add 256 to the number you have just used and input this number into the program. This time you will notice a difference as you are now using the new colours. **The second group of colours from 256 to 511 are the 256 palette colours.**

We are going to skip over the 3rd group of colours as they cannot be illustrated by this program. These are the colours from 512 to 767 and it is better to avoid these altogether until you have gained confidence in using the other groups. These colours modify the appearance of most pointer environment programs.

Now add 768 to the number you have noted down and enter this. **The group of colours from 768 to 1023 are various shades of grey.** (Aurora users may have some difficulties with this part of the tutorial)

Let us make a summary of the groups of colours so far:

0 - 255	**QL Colours**
256 - 511	**256 Palette Colours**
512 - 767	**System Colours (for experts only!)**

768 - 1023	**Grey shades**

We still have another three groups to do:

1024 - 1279	**3D Borders**
16384- 32767	**Stipples**
32768 - 65535	**RGB Colours**

To examine the 3D borders we have to do something different. We have to break into our basic program again and amend line 50 so that it reads: 50 WM_BORDER #2,2,1024

Run the program and you will see the border has changed to give the window a slightly raised appearance. Break into the program again and change 1024 to 1025 and when you run it this time you will see the border gives the window a slightly sunken appearance.

The group of colours 1024 to 1279 are the way in which programmers give their programs a 3D appearance. For obvious reasons you should only use these numbers with the WM_BORDER command.

Please note the colours of the border are determined by the colours your version of SMSQ/E uses to display the pointer environment. If you want to use your own colours, you have to instruct the window manager to use these colours, set the 3D border and then instruct the window manager to revert to the old colours. This is simple to do but beyond the scope of this tutorial. I have not done a great amount of research on 3D effects, but one person who has, Wolfgang Uhlig, advises the use of the range 1024 to 1039

We now move into the realm of the big numbers. Try running the program with numbers in the range 16384 to 32767. These are stippled colours. I shall describe these in some detail in the last part of this tutorial.

The last group of colours are the RGB colours and they are in the range 32768 to 65535. Again you can experiment with several numbers in this range. Aurora card users should remember however that, although there are 16,383 stipples and 32767 RGB colours, they cannot display the majority of these.

You now know how to use the new colours in your own programs without any knowledge of pointer environment programming.

First the easy bit. To use the new colours all you have to do is **change PAPER, INK, STRIP BLOCK and BORDER to WM_PAPER, WM_INK, WM_STRIP WM_BLOCK and WM_BORDER respectively**. When you do this, but don't change the values, the program will look just the same as before because the first 256 Window Manager colours are the same as the old QL ones. You can change the values of the colours at your leisure.

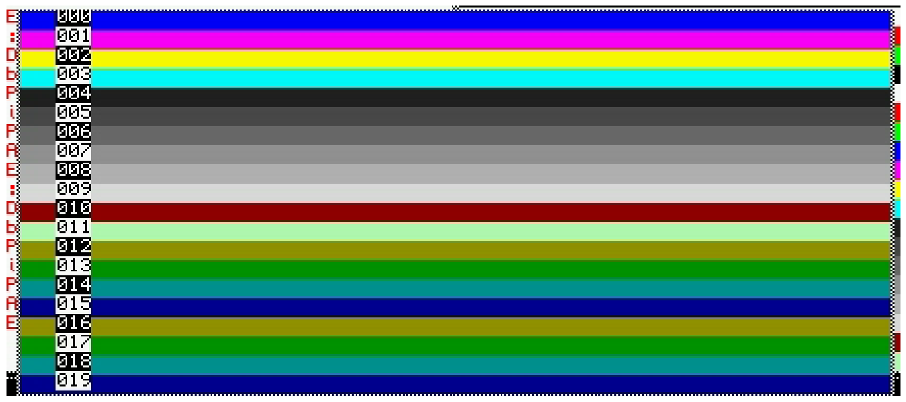
CHOOSING COLOURS

Now the difficult bit. How do you know which number to use for which colour?

There are a number of tools to help you.

Firstly the 256 palette colours. In the SMSQ/E manual you will find a short program to display these colours. To save you looking it up it is reproduced below in a slightly amended form to make it easier to compile. You can scroll through the colours by pressing the up and down arrow keys. Don't forget that to use these colours in your programs you will have to add 256 to the numbers shown against the colour.

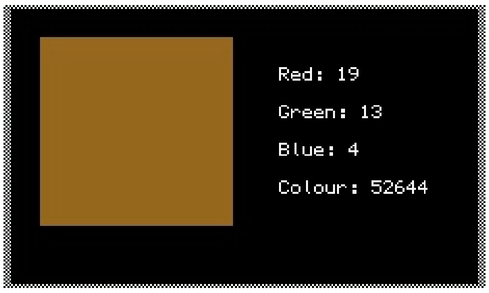
```
100 OPEN #0,con : out=0
110 WINDOW #out, 16*10+2, 16*10+2,50,50
120 COLOUR_PAL : BORDER #out,1,0,1
130 bottom=-16
140 FOR i= 1 TO 16 : up
150 REPeat up_down
160 BGET #0,a
170 IF a=208 : IF bottom<255-16 : up
180 IF a=216 : IF bottom>0 : down
190 END REPeat up_down
200 :
210 DEFine PROCedure up
220 bottom=bottom + 1
230 PAPER bottom + 15 :SCROLL -10
250 i=bottom+1015 : I$=i
270 PAPER #out, i&&1 : INK #out, (i+1) &&1
280 AT #out,15,0 : PRINT #out,I$(2 TO 4)
290 END DEFine
300 :
320 DEFine PROCedure down
330 bottom=bottom-1
340 PAPER bottom  : SCROLL 10
350 i=bottom + 1000 : I$=i
380 PAPER #out, i&&1 : INK #out, (i+1) &&1
390 AT #out,0,0 : PRINT #out,I$(2 TO 4)
400 END DEFine
```



```
000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
```

Now the RGB colours. Below is a short basic program I wrote for my own use when I was learning about the new colours. What I like about this program is that it helps you to learn how the colours are made up from the Red, Green and Blue components. Press the r g, and b keys to increase the proportion of red, green and blue respectively. To reduce the amount of each colour press shift plus r g or b. The up arrow key increases all colours together and the down arrow key reduces it.

```
100 WINDOW 250,150,0,0
110 BORDER 2,248
120 PAPER 0 : INK 7 : CLS
130 red=0 : green=0 blue=0
140 print_colours
150  REPeat change
160  e$=INKEY$(-1)
170  e=CODE(e$)
180  SELect ON e
190   = 114 : REMark r key - increase red
200     red = red + 1
210   = 72 : REMark Shift+r key - decrease red
220     red = red - 1
230   = 103 : REMark g key - increase green
240     green = green +1
250   = 71 : REMark Shift+g key - decrease green
260     green = green - 1
270   = 98 : REMark b key - increase blue
280     blue = blue + 1
290   = 66 : REMark shift+b key decrease blue
300     blue = blue - 1
310   = 208 : REMark Up cursor key - increase brightness
320     red = red +1 : green = green +1 : blue = blue +1
330   = 216 : REMark     Down cursor key - decrease brightness
340     red = red -1 : green = green -1 : blue = blue -1
350  END SELect
360  IF red>30 : red=31
370  IF green>30 : green=31
380  IF blue>30 : blue=31
390  IF red<1 : red=0
400  IF green<1 : green=0
410  IF blue<1 : blue=0
420  print_colours
430 END REPeat change
440      :
450 DEFine PROCedure print_colours
460  AT 3,23 : PRINT “Red: “;red:”;” “
470  AT 5,23 : PRINT “Green: “;green;”;” “
480  AT 7,23 : PRINT “Blue: “;blue;” ;” “
490  Colour=32768 + 1024*red + 32*green + blue
500  AT 9,23 : PRINT “Colour: “;Colour
510  WM_BL0CK l00,100,15,15,Colour
520 END DEFine
```

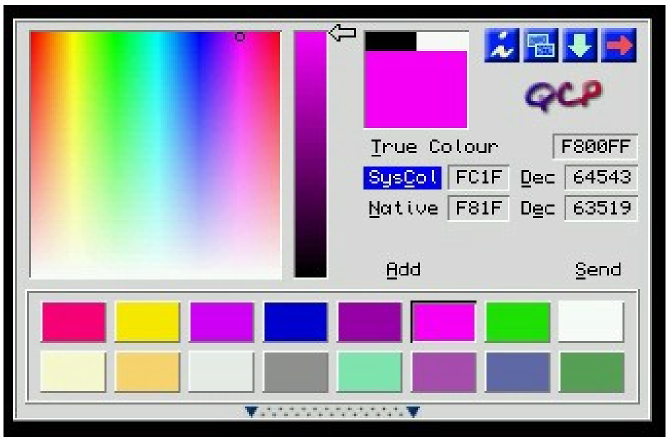


```
Red: 19
Green: 13
Blue: 4
Colour: 52644
```

However if you want to do **choose colours in a more professional way**, the program to use is

QCP which stands for QL Colour Picker by Wolfgang Uhlig and Bob Spelten. You can download the program from this website:

http://members.chello.nl/b.spelten/ql/



If you look at the screen shot of this program. You will see a large window on the left hand side with a cursor in the shape of a circle at the top right hand side of the window. You can move this cursor around to choose the colour you want, which is displayed in the square just right of centre. If necessary you can adjust the brightness of the colour by the strip next to the arrow. SysCol (or System Colour) gives you the number you have to input to get that colour in your program and you can put this in the stuffer buffer to be dropped in your program when you press ALT + SPACE

It is worthwhile to check if you have the latest copy of QCP because the early versions of the program only gave values in HEX, but the latest version gives then in both HEX and DECIMAL. Using the value for the purple colour shown in the screenshot you could enter this into your program either as WM_PAPER 64543 or WM_PAPER $FC1F.

You now have all you need to know to experiment with the new colours in your own programs. Perhaps it is a good idea to pause here and come back to the Stipple Colours later.

STIPPLE SUPPLEMENT

So far we looked at GD2 colours using a short SuperBasic program:

```
10 WTV
20 INK 7 : PAPER 0 : CLS
30 WINDOW #2, 300,154,50,50
40 PAPER #2, 0 : CLS #2
50 WM_BORDER #2,2,1025
60 AT 2,3 : INPUT "colour "colour'
70 WM_BLOCK #2, 292,50,0,0, colour
80 WM_BLOCK #2, 292,50,0,50,colour+5
90 WM_BLOCK #2, 292,50,0,100, colour+10
100 PAUSE
110 GO TO 60
```

We shall be using a modified version of the program to learn about stipples.

This is the magic formula the QL uses to display GD2 stipples:

%01ssxxxxxxyyyyyy

This represents two bytes and it is a little easier to follow if we split it into the separate bytes:

01ssxxxx xxyyyyyy

The lowest number of the first of these two bytes is 01000000 or 2 x 2 x 2 x 2 x 2 x 2 x 2 = 64. The highest number is 01111111 or 128. As this is the higher of the two bytes we have to multiply these numbers by 256 to give 16384 and 32767 respectively In other words the start of the first byte is telling the QL we have a number between 16384 and 32767 which is the stipple range.

The two letters ss tell the computer the sort of stipple:

0 = dot
1 = horizontal stripe
2 = vertical stripe
3 = checkers

In simple language dots will be in the range 16384 to 20479; horizontal stripes in the range 20480 to 24575; vertical stripes in the range 24576 to 28671; and checkers in the range 28672 to 32767.

We can begin to build up a formula more simple for humans to understand:

stipple = 16384 + type*4096 + colour information.

We now come to the xxxxxx. This is the first colour and there are no prizes for guessing that yyyyyy is the second colour

There are six x's so the maximum number of colours is 2 x 2 x 2 x 2 x 2 x 2 = 64. Similarly there are six y's and the maximum number of colours is 64.

We can now complete the formula that humans can understand:

stipple = 16384 + type*4096 + colour1*64 + colour2

Type can only be 0, 1, 2 or 3 and each colour can have a value between 0 and 63.

To experiment with stipple colours our short basic program is very similar to the program we used to look at GD2 colours in general:

```
10 WTV
20 INK 7 : PAPER 0 : CLS
30 WINDOW #2, 300,124,50,50
40 PAPER #2, 0 : CLS #2
50 WM_B0RDER #2,2,1025
60 AT 1,3 : PRINT "              "
70 AT 2,3 : PRINT "              "
```

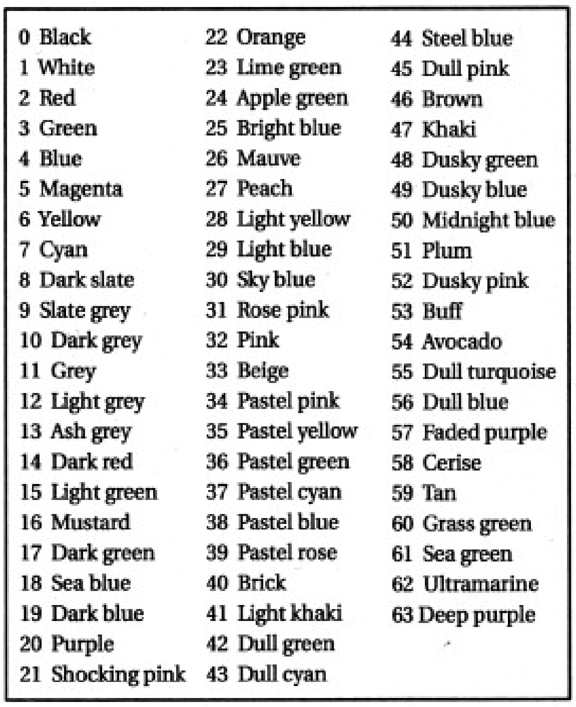
```
80 AT 1,3 : INPUT “1st colour “; colourl
90 AT 2,3 : INPUT “2nd colour “;colour2
100 colour = 16384 + 64*colour1 + colour2
110 WM_BL0CK #2,292,30,0,0 colour : REMark dot
120 AT 5,5 : PAPER 0 : INK 7 : PRINT colour
130 WM_BLOCK #2, 292,30,0,30,colour + 4096 : REMark horizontal
stripe
140 AT 8,5 : PAPER 0 : INK 7  : PRINT colour + 4096
150 WM_BLOCK #2, 292,30,0,60,colour + 8192 : REMark vertical
stripe
160 AT 11,5 : PAPER 0 : INK 7 : PRINT colour + 8192
170 WM_BL0CK #2, 292,30,0,90, colour + 12288 : REMark checkers
180 AT 14,5 : PAPER 0 : INK 7 : PRINT colour + 12288
190 PAUSE
200 GO TO 60
```

If you already have the original program loaded into your machine, you can save time by modifying it.

First of all there is a slight change to the size of the window defined in line 30. As we are now working with two colours there are some additional input lines (60 to 90). After the input information there is the formula for calculating stipple colours (100). Then there is an additional WM_BLOCK command and also changes to the size of the existing blocks. Finally after each block command there is a line to print the stipple number.

We now have a program for viewing stipples, but there is still a practical problem of knowing which colours are actually used in the stipples. They are the first 64 colours in the 256 colour palette. Fortunately they have been given names and you can look these up in the table below.

So if we want a red and white stipple, we could say colour1 = 1 and colour2 = 3. Try inserting various values in the short program you have typed in. You will find you can get some interesting backgrounds for your programs by forming stipples from two of the pastel shades.



0 Black	22 Orange	44 Steel blue
1 White	23 Lime green	45 Dull pink
2 Red	24 Apple green	46 Brown
3 Green	25 Bright blue	47 Khaki
4 Blue	26 Mauve	48 Dusky green
5 Magenta	27 Peach	49 Dusky blue
6 Yellow	28 Light yellow	50 Midnight blue
7 Cyan	29 Light blue	51 Plum
8 Dark slate	30 Sky blue	52 Dusky pink
9 Slate grey	31 Rose pink	53 Buff
10 Dark grey	32 Pink	54 Avocado
11 Grey	33 Beige	55 Dull turquoise
12 Light grey	34 Pastel pink	56 Dull blue
13 Ash grey	35 Pastel yellow	57 Faded purple
14 Dark red	36 Pastel green	58 Cerise
15 Light green	37 Pastel cyan	59 Tan
16 Mustard	38 Pastel blue	60 Grass green
17 Dark green	39 Pastel rose	61 Sea green
18 Sea blue	40 Brick	62 Ultramarine
19 Dark blue	41 Light khaki	63 Deep purple
20 Purple	42 Dull green	
21 Shocking pink	43 Dull cyan	