The Risc PC Column

by Lee Calcraft

Text in 24-bit Colour

In the August/September issue, we experimented with 24-bit graphics in Basic. It is now the turn of text in fancy fonts. The arrival of 24-bit colours for fancy fonts marks a true milestone in text presentation on the Acorn 32-bit range. The reason for this lies in the nature of anti-aliasing. This is a technique used by Acorn, and more recently by other manufacturers, in which the ragged edges of curves (and in particular the borders of text characters) are made to appear smooth by inserting pixels of intermediate shade at appropriate positions along each edge.

The Acorn implementation can make use of shade ranges of up to 16 components, giving a very smooth look to their anti-aliased text. Up to now however, it was very difficult to find a range of 14 colours intermediate between a foreground and background colour. In fact, even in the 256-colour modes this was only possible for white text on black (or the reverse). But now with a palette of up to 16 million colours, you can anti-alias text in any colour against any background, and perform it to a very high degree of precision. The result is that you can create screens which are tasty enough to eat.

But how is it done? Well, it takes four or five system calls to set up fonts and colours, and then to write them on the screen - just as it does with 256 colours. But once you have got the hang of it, and better still, knocked up a couple of functions to make the necessary calls, it s a breeze.

Here is a typical sequence of calls:

```
SYS "Font_FindFont",,"Homerton.BOLD",x,y TO my_font
SYS "Font_SetPalette",,,n,14,bgnd%,fgnd%, &65757254
SYS "Font_SetFont",my_font
SYS "Font_SetFontColours",,,col,14
SYS "Font_Paint",,text$,20,X,Y
```

The first sets up Homerton bold in the required size, and assigns it to the font handle my_font. The second sets up an anti-aliasing palette. This is the only call which has changed with the introduction of 24-bit graphics, and it is now easier to use than before because we can pass it full 24-bit colour data. Font_SetFont simply sets our chosen font as the current font, while Font_SetFontColours does a similar thing for the anti-alias palette. Finally Font_Paint actually puts the text on the screen.

Tasty enough to eat - well almost

The accompanying listing gives a practical example, and generates blue text on a light green background. If when you run this program, the curves on the letters seem to be a bit jagged, you probably need to set the Anti-alias up to parameter. To do this, double-click on !Boot in the root directory to display the control panel, then click on Fonts, and increase the value at the top of the Fonts display to

100, say.

It is probably worth experimenting a little with the program to discover the strengths and limitations of the new colour-rich anti-aliasing. To change the background colour, just alter the values of br, bg, and bb - the rgb components. Similarly, fr, fg, and fb determine the foreground colour. Each component should lie in the range 0-255.

Note the use of the procedures PROCfont_colour, which sets up the anti-alias palette, and PROCwrite, which puts the text on the screen.

Shadowed Text

The massively extended range of colours available on the Risc PC also makes it possible to implement shadowed text more effectively. It is very tempting to think that the way to create a shadow is to choose a suitable offset (of a few OS units), and simply write the text in black at the offset, and then in the normal text colour at the required position. And while this works for the system font, it does not look very good when used with anti-aliased fonts. The reason is twofold. Firstly, shadows are rarely completely black, but more importantly, the anti-aliased pixels show up against the black shadow, destroying the effect completely.

However, you can largely avoid this by choosing a shadow colour which is just a little darker than the background. In this way the anti-aliased pixels do not stand out too much, and a realistic shadowing effect can be achieved.

To add shadows to the current program, just insert the following lines immediately after the OFF statement early in the program:

```
s=40
s_x=10
s_y=7
PROCfont_colour(pal_no,br-s,bg-s,bb-s,br,bg,bb)
PROCwrite(text$,font,pal_no,100+s_x,800-s_y)
```

All that we have done is to replace the foreground rgb components with those of the background, but with each reduced by a factor of s. Making s greater than 40 will darken the shadow. We have then offset the printing position by s_x and s_y, using values of 10 and 7 respectively. As always however, the best results are found by experiment.

```
 10 REM Program   Risc PC Font tests
 20 REM Version   A 1.00
 30 REM Author    Lee Calcraft
 40 REM RISC User November 1994
 50 REM Program   Subject to Copyright
 60 REM           Not Public Domain
 70 :
 80 ON ERROR:OSCLI("set DeskEdit$ERL "
+STR$(ERL)):PRINT TAB(0,0);REPORT$;" at
line ";ERL:END
 90
 100 REM MODE "X800,Y600,C32K,EX1,EY1"
 110 MODE 28
 120
 130 xpoint=80:ypoint=80
 140 text$="The Risc PC"
 150 pal_no=1
 160
 170 SYS "Font_FindFont",,"Homerton.BOL
D",xpoint*16,ypoint*16 TO font
 180 REM Background
 190 br=120        :REM red
 200 bg=180        :REM green
 210 bb=120        :REM blue
 220
 230 REM Foreground
 240 fr=0          :REM red
 250 fg=0          :REM green
 260 fb=200        :REM blue
 270
 280 GCOL 128,br,bg,bb:CLG
 290 OFF
 300
 310 PROCfont_colour(pal_no,fr,fg,fb,br
,bg,bb)
 320 PROCwrite(text$,font,pal_no,100,80
0)
 330
 340 END
 350
 360 DEF PROCwrite(text$,font,col,X,Y)
 370 SYS "Font_SetFont",font
 380 SYS "Font_SetFontColours",,,col,14
 390 SYS "Font_Paint",,text$,20,X,Y
 400 ENDPROC
 410
 420 DEF PROCfont_colour(n,r1,g1,b1,r0,
g0,b0)
 430 bgnd%=FNconvert(r0,g0,b0)
 440 fgnd%=FNconvert(r1,g1,b1)
 450 SYS"Font_SetPalette",,,n,14,bgnd%,
fgnd%,&65757254
 460 ENDPROC
 470
 480 DEF FNconvert(r,g,b)
```