

## Re: richard heathfield

**Source:** <http://coding.derkeiler.com/Archive/General/comp.programming/2003-11/1302.html>

---

**From:** Arthur J. O'Dwyer ([ajo\\_at\\_nospam.andrew.cmu.edu](mailto:ajo_at_nospam.andrew.cmu.edu))

**Date:** 11/18/03

Date: Mon, 17 Nov 2003 18:19:52 -0500 (EST)

On Mon, 17 Nov 2003, Mantorok Redgormor wrote:

>  
> *Richard Heathfield wrote...*  
> > *Mantorok Redgormor wrote:*  
> > >  
> > > *Well I am still dying to know, how did you create that*  
> > > *ascii art program? What is the technique you used?*  
> >  
> > *Er, let me think...*  
> >  
> > *Okay, I started off by taking a photograph of Clint, my debugging*  
> > *teddybear. The photograph was in JPEG format. I transferred it to*  
> > *my Linux box, where I used the GIMP to turn it into Windows bitmap*  
> > *format (basically because bitmaps are easier to code for). I then*  
> > *flipped it back over to Windows, and wrote a C++ Builder program to*  
> > *load the bitmap into a canvas for me. That gave me pixel-by-pixel*  
> > *access. Pause.*  
>  
> *I can convert it to (greyscale)bmp*

Richard apparently used (one of the) RGB BMP format(s).  
But greyscale works, given that ASCII art is greyscale. :)

> *and then run it through hexdump*  
> *not sure if this is good enough*

No; you need to get the actual pixel values involved, not just the bytes stored in the disk file. The most common formats do have a one-to-one correspondence between bytes and pixels (plus a file header), but easier to just load the image up and manipulate the "pixel array" directly.

Blatant plug:

<http://www.contrib.andrew.cmu.edu/~ajo/free-software/ImageFmtc.h>

> > *I also wrote another C++ Builder program. This one wrote the entire*  
> > *printable subset of the character set onto a canvas, using a sensible*  
> > *fixed-pitch font such as might be displayed by a console. It then did*

- > > *pixel-counting to determine the "density" of each letter.*
- >
- > *Do you know of some documentation that explains the above in depth?*

I could tell you how to do it in old MS-DOS, using the system ROM default font. These days, you might see how your user is actually going to view your image: e.g., in black-on-white Courier font. Then figure out the format of TTF font images and proceed as above. But that's hella complicated, IMHO. :-)

- > > *Back to the first program. :-) Okay, I iterated over the bitmap, using*
- > > *square blocks of pixels, and averaging their red, green, and blue values to*
- > > *get a final greyscale value. The program then decided which letter best*
- > > *approximated that value (it wasn't difficult:  $\text{letter}[\text{greyscalevalue} / 4]$ ,*
- > > *basically). And then it was all over bar the shouting. The shouting, in*
- > > *this case, consisted of generating an ever-so-slightly non-obvious but 100%*
- > > *standard ISO C program to display the actual characters.*
- >
- > *likewise for documentation. I am interested in doing this.*

```
for each row in the image
  for each pixel in the row
    display Lookup(pixel value)
```

The details of 'Lookup' depend on how fancy you want to get. A simple version of 'Lookup' would be

```
char Lookup(int p) {
  return (p <= 127)? (' ': ('M'));
}
```

A slightly more complicated (2-bit greyscale instead of 1-bit) version would be

```
char Lookup(int p) {
  char lu[] = " ;fM";
  return lu[p/64];
}
```

Exercise: Extend this method to 8-bit greyscale.

Another way to do it would be to pass 'Lookup' whole blocks of pixels, 2x2 or 4x3 blocks (depending on how high-resolution or incredibly tiny you want to make your ASCII art images):

```
for each pair of rows in the image
  for each pair of pixels in the rows
    display Lookup(top left pixel, top right pixel, bottom ...)
```

```
char Lookup(int p1, int p2, int p3, int p4)
{
```

```
char lu[] = "\\\"-!~r--!7=t19.{/F:IfP+]j9zHZP"  
          ".\\}T+L[Y;5!TxSH9_LJXcL6EjbdGm6dM"  
int ofs = 0;  
ofs |= (p1 > 127) << 3;  
ofs |= (p2 > 127) << 2;  
ofs |= (p3 > 127) << 1;  
ofs |= (p4 > 127) << 0;  
return lu[ofs];  
}
```

This is where the serious font-processing comes in, the stuff that I could tell you about for DOS but not Windows and certainly not for Linux.

Everything else is just sugarcoating.

[Historical interest: Before ASCII art, there was "typewriter art," that was meant to be typed on a typewriter (including the backspace action, for example). Anyone know if anyone ever tried multicolored typewriter art -- one text per primary color?]

-Arthur