

---

## Computer Modern Roman fonts for ebooks

Martin Ruckert

### How it all started

Last year on February 19, I looked at the first version of my first ebook and I was shocked. I had just finished the printed version of “The MMIX Supplement for The Art of Computer Programming” [5] and Donald Knuth had provided extensive help to make its appearance match the books in his series. Donald Knuth had developed  $\text{\TeX}$ , METAFONT, and the Computer Modern Roman (CMR) type faces especially to be able to typeset “The Art of Computer Programming” [3] in the best possible quality. But when I looked at my newly bought Kindle Paperwhite—not the most expensive, but still a decent ebook reader—what I saw (Figure 1) did not resemble even remotely what you would expect from  $\text{\TeX}$  and friends.

I studied the specification of the epub format and found that TrueType or OpenType fonts should work with it. My first attempt with the TrueType versions of the CMR fonts failed because I had overlooked the few instances where the characters in the book were not pure ASCII. So I switched to the Computer Modern Unicode (CMU) version of the fonts [4], and mailed the publisher the first long list of change requests including the request to use these fonts. When I received the next version of my ebook, Dayna Isley, the digital development editor responsible for the ebook, wrote: “I’m finding that embedded fonts are not well supported across Kindle apps and devices. In most cases, the fonts default to the standard Kindle fonts. Kindle for PC and Paperwhite support embedded fonts, but the body font is difficult to read (very faint) and therefore not effective.” And see for yourself (Figure 2), she was right.

My schedule was tight, I was teaching 18 credit hours that semester, and aside from the fonts there were more and bigger problems to be solved before the ebook could be released. So we settled for a selection of standard ebook fonts and moved on. The final ebook uses Baskerville fonts for the main text body. It is no match for the printed version, but it was a good compromise given the limitations of time and technique.

Now, a year later, I decided to get back to the problem of ebook production with more time to my disposal: I plan to use my sabbatical in 2017 to build a prototype ebook renderer that uses the algorithms of  $\text{\TeX}$  for ebook layout and a front-end

The multiplicative hashing scheme is equally easy to do, but it is slightly harder to describe because we must imagine ourselves working with fractions instead of with integers. Let  $w$  be the word size of the computer, so that  $w$  is usually  $2^{32}$  or  $2^{64}$  for MMIX; we can regard an integer  $A$  as the fraction  $A/w$  if we imagine the radix point to be at the left of the word. The method is to choose some integer constant  $A$  relatively prime to  $w$ , and to let

$$h(K) = \left\lfloor M \left( \left( \frac{A}{w} K \right) \bmod 1 \right) \right\rfloor. \quad (4)$$

Fig. 1: First version of my ebook

The multiplicative hashing scheme is equally easy to do, but it is slightly harder to describe because we must imagine ourselves working with fractions instead of with integers. Let  $w$  be the word size of the computer, so that  $w$  is usually  $2^{32}$  or  $2^{64}$  for MMIX; we can regard an integer  $A$  as the fraction  $A/w$  if we imagine the radix point to be at the left of the word. The method is to choose some integer constant  $A$  relatively prime to  $w$ , and to let

$$h(K) = \left\lfloor M \left( \left( \frac{A}{w} K \right) \bmod 1 \right) \right\rfloor. \quad (4)$$

Fig. 2: Second version of my ebook

that translates  $\text{\TeX}$  input to an intermediate representation that can be used by the new rendering engine. In preparation for this project, I started to identify those subproblems which I would need to ignore in order to have a reasonable sized project. This brought me back to investigating the font issue.

### ebook versus Preview

One of the good programs to view  $\text{\TeX}$  output on-screen is YAP, which comes with MiK $\text{\TeX}$  [6]. YAP is an acronym standing for Yet Another Previewer. The word “previewer” indicates that it is the purpose of the program to give the user an advance view, an approximation of what one should expect to see on paper. The paper version is the “real thing” and the electronic version is only an intermediate step in its production. An indication of this attitude is the selection of fonts. In its default configuration, YAP uses METAFONT in `ljfour` mode to generate bitmap fonts. These bitmaps are optimized for a (once) popular 600 dpi laser printer. YAP scales them down to display the  $\text{\TeX}$  output on the low resolution computer screen, and if you reduce down-scaling, you can see precisely, down to the last pixel, what you are supposed to get on paper.

The situation has changed significantly with the introduction of ebooks. Now books are produced

specifically for reading on some kind of computer screen. The electronic rendering is no longer an approximation of something yet to come, it is the final product.

The built-in font rendering software (and hardware) of computers usually does not support METAFONT generated bitmapped fonts. It supports TrueType or OpenType outline fonts. These are now the de facto standards. The emerging universal standard in character encoding is, perhaps due to the World Wide Web, the UTF-8 encoding. Outline fonts can be scaled to any resolution desired, but as we will shortly see, this is not sufficient for optimal on-screen reading.

### Rendering Computer Modern Roman fonts

To investigate the rendering of the available CMR fonts on current electronic devices, a reference rendering is needed. I choose (somewhat but not completely arbitrarily) my personal copy of *The METAFONTbook* and picked the second paragraph of the preface [2, page v]. It reads, typeset below in 10pt Computer Modern Roman as in the printed book:

“ Modern printing equipment based on raster lines—in which metal “type” has been replaced by purely combinatorial patterns of zeroes and ones that specify the desired position of ink in a discrete way—makes mathematics and computer science increasingly relevant to printing. We now have the ability to give a completely precise definition of letter shapes that will produce essentially equivalent results on all raster-based machines. Moreover, the shapes can be defined in terms of variable parameters; computers can “draw” new fonts of characters in seconds, making it possible for designers to perform valuable experiments that were previously unthinkable.”

I then took a photograph of this paragraph from the book (Figure 3) which I will use as my reference for the Computer Modern Roman 10pt font from now on. Apart from the plain text, the photograph contains an insert with the first two letters magnified six times. In comparing the different font renderings, one should pay special attention to the thickness of the different strokes of the “M” and the rounding of the “o”. As a first observation you might notice that the font as shown in figure 3 appears to be much heavier than the same font in the previous paragraph—depending of course on how you printed or rendered this article in order to read it. While it is impossible for me to avoid the effects that your rendering software and your output device

Modern printing equipment based on raster has been replaced by purely combinatorial patterns ify the desired position of ink in a discrete way—nputer science increasingly relevant to printing. We r completely precise definition of letter shapes that w alent results on all raster-bas in terms of variable paramete in seconds, making it possible were previously unthinkable.




Fig. 3: From *The METAFONTbook*

Modern printing equipment based on raster has been replaced by purely combinatorial patterns ify the desired position of ink in a discrete way—nputer science increasingly relevant to printing. We n completely precise definition of alent results on all raster-based in terms of variable parameters; seconds, making it possible for were previously unthinkable.




Fig. 4: CMU font, ebook

Modern printing equipment based on raster has been replaced by purely combinatorial patterns ify the desired position of ink in a discrete way—nputer science increasingly relevant to printing. We n completely precise definition of letter shapes that w alent results on all raster-based in terms of variable parameters; seconds, making it possible for were previously unthinkable.




Fig. 5: LM font, ebook

will have on the appearance of fonts, I can reasonably hope that the reproduction of the photographs shown in this article preserve the relative differences which I observed on my output devices.

I have tried my best to take all the photographs in this article under identical conditions, for the sake of comparison. Using a good camera (Canon EOS 60 D, 18Mpixel, 18mm–135mm lens, 1/15s, 5300K, ISO320, 56mm focal length, 16 aperture), I took all photographs under identical light conditions and post-processed the raw images in the same way, trying to reproduce the differences in appearance as well as possible.

### TrueType, OpenType, and Unicode fonts

As a first example, lets look at the rendering on my ebook reader (Kindle Paperwhite 2, 1024x758 pixels, 212dpi) using the OpenType Computer Modern

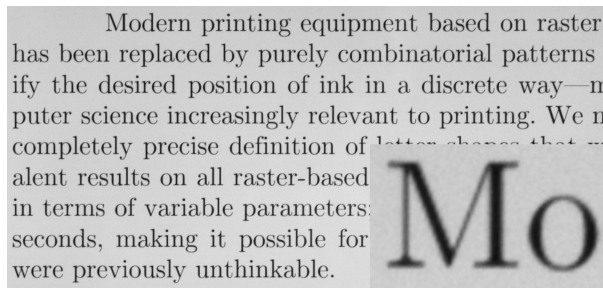


Fig. 6: CMU Font, smart-phone

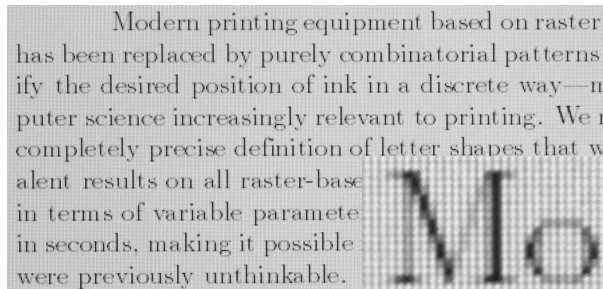


Fig. 7: CMU Font, laptop

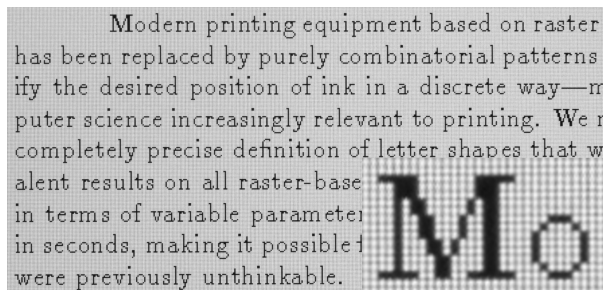
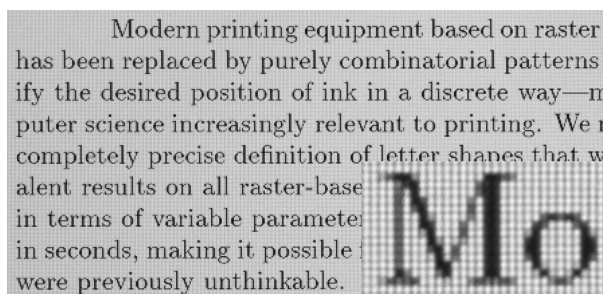
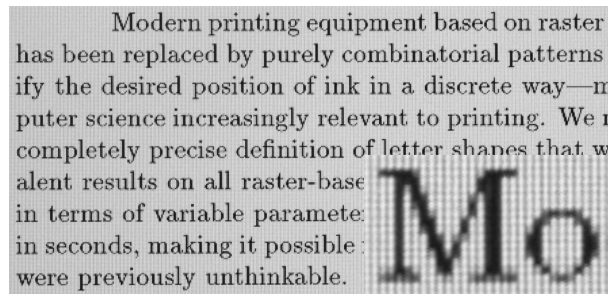
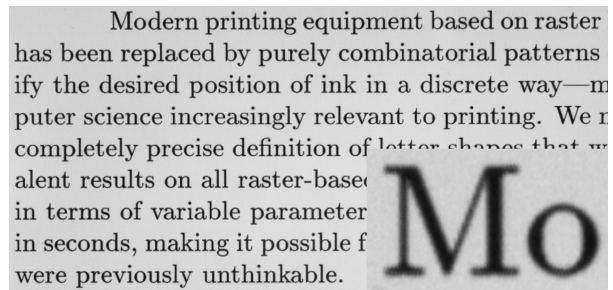
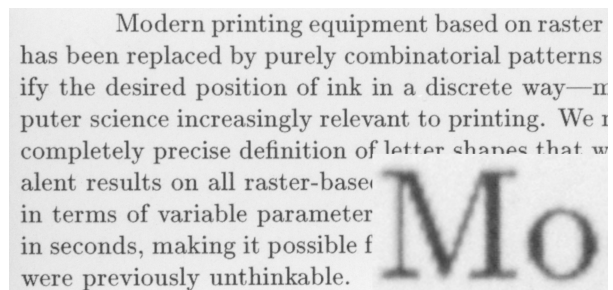


Fig. 8: METAFONT at 142dpi, laptop

Fig. 9: METAFONT at  $4 \times 142$ dpi, laptop

Unicode (CMU) fonts that I had tried already for my ebook (Figure 2). Figure 4 shows how the ebook renders this outline font using the built-in rendering engine. Comparing it with the printed book (Figure 3), it is obvious that the rendering lacks contrast and looks significantly lighter. It turns out that the initial observation that the font is “very faint” is not a general property of the CMR fonts but a property

Fig. 10: METAFONT with *blacker* = 0.6, laptopFig. 11: METAFONT with *blacker* = 1.6, smart-phoneFig. 12: METAFONT with *blacker* = 2.4, ebook

of a specific font implementation on a specific output device. Another popular choice are the OpenType Latin Modern (LM) fonts [1]. The native rendering on the Kindle Paperwhite (Figure 5) is comparable to that of the CMU fonts. It seems that the eInk technology used on the Kindle Paperwhite just needs heavier fonts.

When considering reading text on an electronic device, two other choices come to mind: laptop computers and smart phones (or tablets), which typically have a smaller screen but higher resolution. Figures 6 and 7 show the reference text as displayed on my smart-phone (Motorola Moto G, 1280x720 pixels, 329dpi) and my laptop (Dell Latitude E6530, 1920x1080 pixels, 142dpi).

It is clearly visible that, due to high resolution and good contrast, the font rendering on the smart-phone already approaches the rendering on traditional paper, whereas the laptop screen falls short of

our expectations. METAFONT was designed to produce good looking fonts at low resolution. Donald Knuth writes: “However, it will always be less expensive to work with devices of lower resolution, and we want the output of METAFONT to look as good as possible on the machines that we can afford to buy.” [2, page 195] Of course, we can hope that the ever-increasing resolution of our computer screens will make those techniques dispensable within the next years. But for the time being and for affordable, low-cost devices, good font rendering will continue to be an issue.

### METAFONT and bitmapped fonts

METAFONT is aware of rasterization and takes great care to round the outlines of the glyphs to the available raster, but it assumes an output device that places small dots of black ink on white paper. In contrast, my ebook is able to produce 16 gray levels and my smart-phone screen is, at least in theory, capable of 256 shades of gray. (Other font rendering engines use even more sophisticated subpixel rendering.) To overcome this limitation of bitmap fonts generated by METAFONT, one can render the bitmaps for a higher resolution and then scale down the result to a lower resolution, converting partially-black regions to gray pixels. The effect of this mechanism can be seen in figure 8 and figure 9. Clearly the downscaling gives superior results. So the following figures all show fonts that are scaled down by a factor of 4.

The METAFONT system for font design offers special parameters to adapt the generated bitmap fonts for any specific output device [2, Chapter 24, Discreteness and Discretion]. The main parameter, of course, is the resolution. Since we are dealing with fonts that are too light, we turn our attention to the parameter *blacker*. The variable *blacker* is a special correction intended to help adapt a font to the idiosyncrasies of the current output device [2, page 93]. Its effect can be seen when comparing figure 9 to figure 10, where the parameter *blacker* has been chosen so that the visual appearance of the font on screen would match as closely as possible the appearance in the printed book (Figure 3). Similar results can be obtained for the smart-phone (Figure 11) and the ebook (Figure 12) with appropriately chosen values of *blacker*. The illustrations show that with appropriate parameters, the glyphs as rendered by METAFONT look better than their counterparts produced by the built-in font rendering engines from standard outline fonts optimized for high-resolution printers.

Martin Ruckert

### Conclusion

Preparation of an ebook from a T<sub>E</sub>X source will always be more than just flipping a switch in the T<sub>E</sub>X file. Just as preparing a book for print is more than just adopting the publisher’s style file: it might require for example stretching a paragraph by rewriting it to get a good page break; repositioning and redesigning illustrations, so that they fall on the right page and fit the available space on the page. These things are no longer necessary nor possible with ebooks, but other problems appear: Now the author has to judge the appearance of tables or program listings at different sizes and optimize font sizes for good readability at various magnification levels. Still, I expect that the algorithms of T<sub>E</sub>X can help us to produce ebooks of much better quality than the ebooks we have today.

But even if I can get T<sub>E</sub>X to produce a beautiful page layout for the ebook reader, I still need — especially for the traditional look and feel of books like “The Art of Computer Programming” — a TrueType or OpenType version of the Computer Modern Roman font family using Unicode encoding that is specifically designed for ebooks (or other on-screen reading). My experiments indicate that such fonts are possible and I sincerely hope that one of the many font specialists takes on this project. If you do, please let me know!

### References

- [1] Bogusław Jackowski and Janusz M. Nowacki. The Latin Modern (LM) Family of Fonts. <http://www.gust.org.pl/projects/e-foundry/latin-modern/>, 2009.
- [2] Donald E. Knuth. *Computers & Typesetting, The METAFONT book*. Addison-Wesley, 1986.
- [3] Donald E. Knuth. *The Art of Computer Programming*. Addison Wesley, 1998.
- [4] Andrey V. Panov. Computer Modern Unicode Fonts. <http://canopus.iacp.dvo.ru/~panov/cm-unicode/>, 2010.
- [5] Martin Ruckert. *The MMIX Supplement: Supplement to The Art of Computer Programming Volumes 1, 2, 3 by Donald E. Knuth*. Addison-Wesley, 2015.
- [6] Christian Schenk. MiK<sub>T</sub>E<sub>X</sub>. <http://www.miktex.org/>, 2016.

◇ Martin Ruckert  
Hochschule München  
Lothstrasse 64  
80336 München Germany  
ruckert (at) cs dot hm dot edu