E-books: Old wine in new bottles

Hans Hagen

1 Introduction

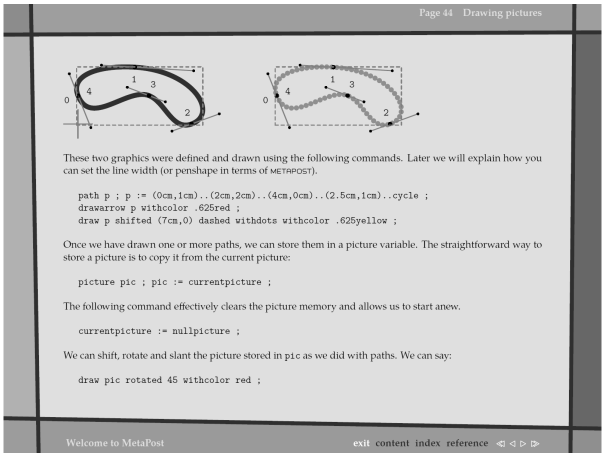
When Dave Walden asked me if TEX (or ConTEXt) can generate ebooks we exchanged a bit of mail on the topic. Although I had promised myself never to fall into the trap of making examples for the sake of proving something I decided to pick up an experiment that I had been doing with a manual in progress and look into the HTML side of that story. After all, occasionally on the ConTEXt list similar questions are asked, like “Can ConTEXt produce HTML?”.

2 Nothing new

When you look at what nowadays is presented as an ebook document, there is not much new going on. Of course there are very advanced and interactive documents, using techniques only possible with recent hardware and programs, but the average ebook is pretty basic. This is no surprise. When you take a novel, apart from maybe a cover or an occasional special formatting of section titles, the typesetting of the content is pretty straightforward. In fact, given that formatters like TEX have been around that can do such jobs without much intervention, it takes quite some effort to get that job done badly. It was a bit shocking to notice that on one of the first e-ink devices that became available the viewing was quite good, but the help document was just some word processor output turned into bad-looking PDF. The availability of proper hardware does not automatically trigger proper usage.

I can come up with several reasons why a novel published as an ebook does not look much more interesting and in many cases looks worse. First of all it has to be produced cheaply, because there is also a printed version and because the vendor of some devices also want to make money on it (or even lock you into their technology or shop). Then, it has to be rendered on various devices so the least sophisticated one sets the standard. As soon as it gets rendered, the resolution is much worse than what can be achieved in print, although nowadays I’ve seen publishers go for quick and dirty printing, especially for reprints.

Over a decade ago, we did some experiments with touch screen computers. They had a miserable battery life, a slow processor and not much memory, but the resolution was the same as on the now fashionable devices. They were quite suitable for reading but even in environments where that made sense (for instance to replace carrying around huge manuals), such devices never took off. Nowadays we have wireless access and USB sticks and memory cards to move files around, which helps a lot. And getting a quality comparable to what can be done today was no big deal, at least from the formatting point of view.



These two graphics were defined and drawn using the following commands. Later we will explain how you can set the line width (or penshape in terms of METAPOST).

```
path p ; p := (0cm,1cm)..(2cm,2cm)..(4cm,0cm)..(2.5cm,1cm)..cycle ;
drawarrow p withcolor .625red ;
draw p shifted (7cm,0) dashed withdots withcolor .625yellow ;
```

Once we have drawn one or more paths, we can store them in a picture variable. The straightforward way to store a picture is to copy it from the current picture:

```
picture pic ; pic := currentpicture ;
```

The following command effectively clears the picture memory and allows us to start anew.

```
currentpicture := nullpicture ;
```

We can shift, rotate and slant the picture stored in pic as we did with paths. We can say:

```
draw pic rotated 45 withcolor red ;
```

Figure 1: A page from the Metafun manual.

If you look in the ConTEXt distribution you will find several presentation styles that can serve as bases for an ebook style. Also some of the ConTEXt manuals come with two versions: one for printing and one for viewing on the screen. A nice example is the Metafun manual (see figure 1) where each page has a different look.

It must be said that the (currently only black and white) devices that use electronic ink have a perceived resolution that is higher than their specifications, due to the semi-analog way the ‘ink’ behaves. In a similar fashion clever anti-aliasing can do wonders on LCD screens. On the other hand they are somewhat slow and a display refresh is not that convenient. Their liquid crystal counterparts are much faster but they can be tiresome to look at for a long time and reading a book on it sitting in the sun is a no-go. Eventually we will get there and I’m really looking forward to seeing the first device that will use a high resolution electrowetting CMYK display.[1] But no matter what device is used, formatting something for it is not the most complex task at hand.

3 Impact

Just as with phones and portable audio devices, the market for tablets and ebook-only devices is evolving rapidly. While writing this, at work I have one ebook device and one tablet. The ebook device is sort of obsolete because the e-ink screen has deteriorated

This article is reprinted from the EuroTEX 2011 proceedings.

[1] http://www.liquavista.com/files/LQV0905291LL5-15.pdf

even without using it and it's just too slow to be used for reference manuals. The tablet is nice, but not that suitable for all circumstances: in the sun it is unreadable and at night the backlight is rather harsh. But, as I mentioned in the previous section, I expect this to change.

If we look at the investment, one needs good arguments to buy hardware that is seldom used and after a few years is obsolete. Imagine that a family of four has to buy an ebook device for each member. Add to that the cost of the books and you quickly can end up with a larger budget than for books. Now, imagine that you want to share a book with a friend: will you give him or her the device? It might be that you need a few more devices then. Of course there is also some data management needed: how many copies of a file are allowed to be made and do we need special programs for that? And if no copy can be made, do we end up swapping devices? It is hard to predict how the situation will be in a few years from now, but I'm sure that not everyone can afford this rapid upgrading and redundant device approach.

A friend of mine bought ebook devices for his children but they are back to paper books now because the devices were not kid-proof enough: you can sit on a book but not on an ebook reader.

The more general devices (pads) have similar problems. I was surprised to see that an iPad is a single user device. One can hide some options behind passwords but I'm not sure if parents want children to read their mail, change preferences, install any application they like, etc. This makes pads not that family friendly and suggests that such a personal device has to be bought for each member. In which case it suddenly becomes a real expensive adventure. So, unless the prices drop drastically, pads are not a valid large scale alternative for books yet.

It might sound like I'm not that willing to progress, but that's not true. For instance, I'm already an enthusiastic user of a media player infrastructure.[2] The software is public, pretty usable, and has no vendor lock-in. Now, it would make sense to get rid of traditional audio media then, but this is not true. I still buy CDs if only because I then can rip them to a proper lossless audio format (FLAC). The few FLACs that I bought via the Internet were from self-publishing performers. After the download I still got the CDs which was nice because the booklets are among the nicest that I've ever seen.

Of course it makes no sense to scan books for ebook devices so for that we depend on a publishing network. I expect that at some point there will be proper tools for managing your own electronic books and in most cases a simple file server will do. And the most open device with a proper screen will become my favourite. Also, I would not be surprised if ten years from now, many authors will publish themselves in open formats and hopefully users will be honest enough to pay for it. I'm not too optimistic about the latter, if only because I observe that younger family members fetch everything possible from the Internet and don't bother about rights, so we definitely need to educate them. To some extent publishers of content deserve this behaviour because more than I like I find myself in situations where I've paid some 20 euro for a CD only to see that half a year later you can get it for half the price (sometimes it also happens with books).

Given that eventually the abovementioned problems and disadvantages will be dealt with, we can assume that ebooks are here and will stay forever. So let's move on to the next section and discuss their look and feel.

4 Interactivity

The nice thing about a paper book is that it is content and interface at the same time. It is clear where it starts and ends and going from one page to another is well standardized. Putting a bookmark in it is easy as you can fall back on any scrap of paper lying around. While reading you know how far you came and how much there is to come. Just as a desktop on a desktop computer does not resemble the average desktop, an ebook is not a book. It is a device that can render content in either a given or more free-form way.

However, an electronic book needs an interface and this is also where at the moment it gets less interesting. Of course the Internet is a great place to wander around and a natural place to look for electronic content. But there are some arguments for buying them at a bookshop, one being that you see a lot of (potentially) new books, often organized in topics in one glance. It's a different way of selecting. I'm not arguing that the Internet is a worse place, but there is definitely a difference: more aggressive advertisements, unwanted profiling that can narrow what is presented to a few choices.

Would I enter a bookshop if on the display tables there were stacks of (current) ebook devices showing the latest greatest books? I can imagine that at some point we will have ebook devices that have screens that run from edge to edge and then we get back some of the appeal of book designs. It is that kind of future devices that we need to keep in mind when we design electronic documents, especially when after

[2] The software and hardware was developed by SlimDevices and currently is available as Logitech Squeezeserver. Incidentally I can use the iPad as an advanced remote control.

some decades we want them to be as interesting as old books can be. Of course this is only true for documents that carry the look and feel of a certain time and place and many documents are thrown away. Most books have a short lifespan due to the quality of the paper and binding so we should not become too sentimental about the transition to another medium.

Once you're in the process of reading a book not much interfacing is needed. Simple gestures or touching indicated areas on the page are best. For more complex documents the navigation could be part of the design and no screen real estate has to be wasted by the device itself. Recently I visited a school-related exhibition and I was puzzled by the fact that on an electronic schoolboard so much space was wasted on colorful nonsense. Taking some 20% off each side of such a device brings down the effective resolution to 600 pixels so we end up with 10 pixels or less per character (shown at about 1 cm width). At the same exhibition there were a lot of compensation programs for dyslexia advertised, and there might be a relationship.

5 Formatting

So how important is the formatting? Do we prefer reflow on demand or is a more frozen design that suits the content and expresses the wish of the author more appropriate? In the first case HTML is a logical choice, and in the second one PDF makes sense. You design a nice HTML document but at some point the reflow gets in the way. And yes, you can reflow a PDF file but it's mostly a joke. Alternatively one can provide both which is rather trivial when the source code is encoded in a systematic way so that multiple output is a valid option. Again, this is not new and mostly a matter of a publisher's policy. It won't cost more to store in neutral formats and it has already been done cheaply for a long time.

Somewhat interfering in this matter is digital rights management. While it is rather customary to buy a book and let friends or family read the same book, it can get complicated when content is bound to one (or a few) devices. Not much sharing there, and in the worst case, no way to move your books to a better device. Each year in the Netherlands we have a book fair and bookshops give away a book specially written for the occasion. This year the book was also available as an ebook, but only via a special code that came with the book. I decided to give it a try and ended up installing a broken application, i.e. I could not get it to load the book from the Internet, and believe me, I have a decent machine and the professional PDF viewer software that was a prerequisite.

6 Using TeX

So, back to Dave's question: if ConTeXt can generate ebooks in the ePub format. Equally interesting is the question if TeX can format an ePub file into a (say) PDF file. As with much office software, an ePub file is nothing more than a zip file with a special suffix in which several resources are combined. The layout of the archive is prescribed. However, by demanding that the content itself is in HTML and by providing a stylesheet to control the renderer, we don't automatically get properly tagged and organized content. When I first looked into ePub, I naively assumed that there was some well-defined structure in the content; turns out this is not the case.

Let's start by answering the second question. Yes, ConTeXt can be used to convert an ePub file into a PDF file. The natural followup question is if it can be done automatically, and then some more nuance is needed: it depends. If you download the ePub for *A tale of two cities* from Charles Dickens from the Gutenberg Project website and look into a chapter you will see this:

```
<h1 id="pgepubid00000">A TALE OF TWO CITIES</h1>
<h2 id="pgepubid00001">A STORY OF THE FRENCH
   REVOLUTION</h2>
<p><br/></p>
<h2>By Charles Dickens</h2>
<p><br/>
<br/></p>
<hr/>
<p><br/>
<br/></p>
<h2 id="pgepubid00002">Contents</h2>
```

What follows is a table of contents formatted using HTML tables and after that

```
<h2 id="pgepubid00004">I. The Period</h2>
```

So, a level two header is used for the subtitle of the book as well as a regular chapter. I must admit that I had to go on the Internet to find this snippet as I wanted to check its location. On my disk I had a similar file from a year ago when I first looked into ePub. There I have:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="en">
 <head>
  <title>I | A Tale of Two Cities</title>
  ....
 </head>
 <body>
  <div class="body">
   <div class="chapter">
    <h3 class="chapter-title">I</h3>
    <h4 class="chapter-subtitle">The Period</h4>
```

I also wanted to make sure if the interesting combination of third and fourth level head usage was still there but it seems that there are several variants available. It is not my intention to criticize the coding, after all it is valid HTML and can be rendered as intended. Nevertheless, the first snippet definitely looks worse, as it uses breaks instead of CSS spacing directives and the second wins on obscurity due to the abuse of the head element.

These examples answer the question about formatting an arbitrary ePub file: “no”. We can of course map the tagging to ConTEXt and get pretty good results but we do need to look at the coding.

As such books are rather predictable it makes sense to code them in a more generic way. That way generic stylesheets can be used to render the book directly in a viewer and generic ConTEXt styles can be used to format it differently, e.g. as PDF.

Of course, if I were asked to set up a workflow for formatting ebooks, that would be relatively easy. For instance the Gutenberg books are available as raw text and that can be parsed to some intermediate format or (with MkIV) interpreted directly.

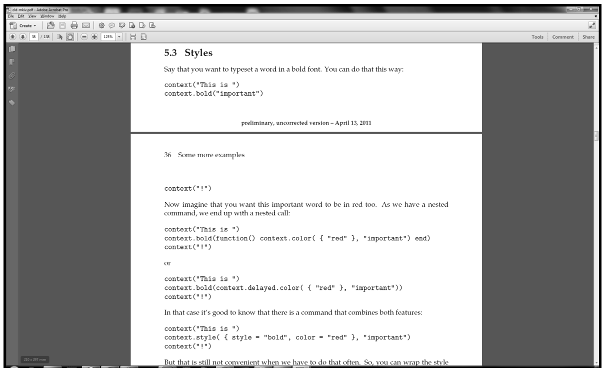
Making a style for a specific instance, like the Dickens book, is not that complex either. After all, the amount of encoding is rather minimal and special bits and pieces like a title page need special design anyway. The zipped file can be processed directly by ConTEXt, but this is mostly just a convenience.

As ePub is just a wrapper, the next question is if ConTEXt can produce some kind of HTML and the answer to that question is positive. Of course this only makes sense when the input is a TEX source, and we have argued before that when multiple output is needed the user might consider a different starting point. After all, ConTEXt can deal with XML directly.

The main advantage of coding in TEX is that the source remains readable and for some documents it’s certainly more convenient, like manuals about TEX. In the reference manual ‘ConTEXt Lua Documents’ (CLD) there are the following commands:

```
\setupbackend
  [export=yes]
\setupinteraction
  [title=Context Lua Documents,
   subtitle=preliminary version,
   author=Hans Hagen]
```

At the cost of at most 10% extra runtime an XML export is generated in addition to the regular PDF file. Given that you have a structured TEX source the exported file will have a decent structure as well and you can therefore transform the file into something else, for instance HTML. But, as we already have a good-looking PDF file, the only reason to have HTML as well is for reflowing. Of course wrapping up the HTML into an ePub structure is not that hard. We can probably even get away from wrapping because we have a single self-contained file.



5.3 Styles

Say that you want to typeset a word in a bold font. You can do that this way:

```
context("This is ")
context.bold("important")
```

preliminary, uncorrected version – April 13, 2011

36 Some more examples

```
context("!")
```

Now imagine that you want this important word to be in red too. As we have a nested command, we end up with a nested call:

```
context("This is ")
context.bold(function() context.color( { "red" }, "important") end)
context("!")
```

or

```
context("This is ")
context.bold(context.delayed.color( { "red" }, "important"))
context("!")
```

In that case it’s good to know that there is a command that combines both features:

```
context("This is ")
context.style( { style = "bold", color = "red" }, "important")
context("!")
```

But that is still not convenient when we have to do that often. So, you can wrap the style

Figure 2: A page from the CLD manual in PDF.

The `\setupbackend` command used in the CLD manual has a few more options:

```
\setupbackend
  [export=cld-mkiv-export.xml,
   xhtml=cld-mkiv-export.xhtml,
   css={cld-mkiv-export.css,mathml.css}]
```

We explicitly name the export file and in addition specify a stylesheet and an alternative XHTML file. If you can live without hyperlinks the XML file combined with the cascading style sheet will do a decent job of controlling the formatting.

In the CLD manual chapters are coded like this:

```
\startchapter[title=A bit of Lua]
\startsection[title=The language]
```

The XML output of this

```
<division detail='bodypart'>
 <section detail='chapter' location='aut:3'>
  <sectionnumber>1</sectionnumber>
  <sectiontitle>A bit of Lua</sectiontitle>
  <sectioncontent>
    <section detail='section'>
      <sectionnumber>1.1</sectionnumber>
      <sectiontitle>The language</sectiontitle>
      <sectioncontent>
```

The HTML version has some extra elements:

```
<xhtml:a name="aut_3">
  <section location="aut:3" detail="chapter">
```

The table of contents and cross references have `xhtml:a` elements too but with the `href` attribute. It’s interesting to search the web for ways to avoid this, but so far no standardized solution for mapping XML elements onto hyperlinks has been agreed upon. In fact, getting the CSS mapping done was not that much work but arriving at the conclusion that (in 2011) these links could only be done in a robust way using HTML tags took more time. (In this example we see the reference `aut:3` turned into `aut_1`. This is done because some browsers like to interpret this colon as a url.) Apart from this the CSS has enough on board to map the export onto something presentable. For instance:

```
sectioncontent {
    display: block ;
    margin-top: 1em ;
    margin-bottom: 1em ; }
section[detail=chapter], section[detail=title] {
    margin-top: 3em ;
    margin-bottom: 2em ; }
section[detail=chapter]>sectionnumber {
    display: inline-block ;
    margin-right: 1em ;
    font-size: 3em ;
    font-weight: bold ; }
```

As always, dealing with verbatim is somewhat special. The following code does the trick:

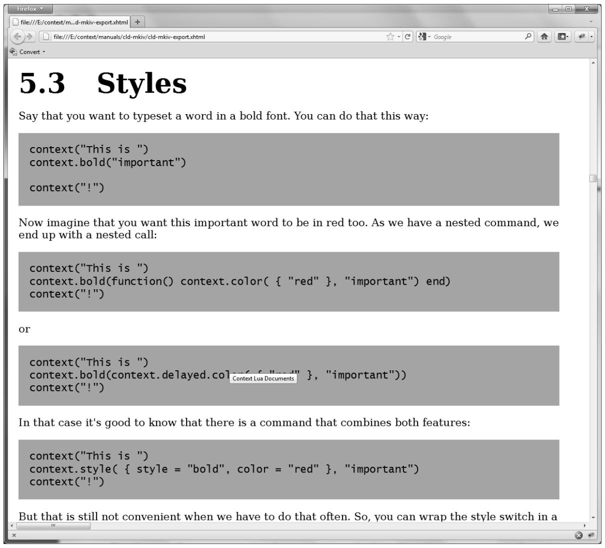
```
verbatimblock {
    background-color: #9999FF ;
    display: block ;
    padding: 1em ;
    margin-bottom: 1em ;
    margin-top: 1em ;
    font-family: "Lucida Console",
      "DejaVu Sans Mono", monospace ; }
verbatimline {
    display: block ;
    white-space: pre-wrap ; }
verbatim {
    white-space: pre-wrap ;
    color: #666600 ;
    font-family: "Lucida Console",
      "DejaVu Sans Mono", monospace ; }
```

The spacing before the first and after the last one differs from the spacing between lines, so we need some extra directives:

```
verbatimlines+verbatimlines {
    display: block ;
    margin-top: 1em ; }
```

This will format code like the following with a bluish background and inline verbatim with its complement:

```
<verbatimblock detail='typing'>
 <verbatimlines>
  <verbatimline>function sum(a,b)</verbatimline>
  <verbatimline> print(a, b, a+b)</verbatimline>
  <verbatimline>end</verbatimline>
 </verbatimlines>
</verbatimblock>
```


Firefox
file:///E:/context/m...d-mkiv-export.xhtml
file:///E:/context/manuals/cld-mkiv/cld-mkiv-export.xhtml
Google
Convert
Context Lua Documents

5.3 Styles

Say that you want to typeset a word in a bold font. You can do that this way:

```
context("This is ")
context.bold("important")

context("!")
```

Now imagine that you want this important word to be in red too. As we have a nested command, we end up with a nested call:

```
context("This is ")
context.bold(function() context.color( { "red" }, "important") end)
context("!")
```

or

```
context("This is ")
context.bold(context.delayed.color( { "red" }, "important"))
context("!")
```

In that case it's good to know that there is a command that combines both features:

```
context("This is ")
context.style( { style = "bold", color = "red" }, "important")
context("!")
```

But that is still not convenient when we have to do that often. So, you can wrap the style switch in a

Figure 3: A page from the CLD manual in HTML.

The hyperlinks need some attention. We need to make sure that only the links and not the anchors get special formatting. After some experimenting I arrived at this:

```
a[href] {
    text-decoration: none ;
    color: inherit ; }
a[href]:hover {
    color: #770000 ;
    text-decoration: underline ; }
```

Tables are relatively easy to control. We have tabulate (nicer for text) and natural tables (similar to the HTML model). Both get mapped into HTML tables with CSS directives. There is some detail available so we see things like this:

```
tablecell[align=flushleft] {
    display: table-cell ;
    text-align: left ;
    padding: .1em ; }
```

It is not hard to support more variants or detail in the export but that will probably only happen when I find a good reason (a project), have some personal need, or when a user asks for it. For instance images will need some special attention (conversion, etc.). Also, because we use MetaPost all over the place that needs special care as well, but a regular (novel-like) ebook will not have such resources.

As an extra, a template file is generated that mentions all elements used, like this:

```
section[detail=summary] {
    display: block ; }
```

with the inline and display properties already filled in. That way I could see that I still had to add a couple of directives to the final CSS file. It also became clear that in the CLD manual some math is used that gets tagged as MathML, so that needs to be covered as well.[3] Here we need to make some decisions as we export Unicode and need to consider support for less sophisticated fonts. On the other hand, the W3C consortium has published CSS for this purpose so we can use these as a starting point. It might be that eventually more tuning will be delegated to the XHTML variant. This is not much extra work as we have the (then intermediate) XML tree available. Thinking of it, we could eventually end up with some kind of CSS support in ConTEXt itself.

It will take some experimenting and feedback from users to get the export right, especially to provide a convenient way to produce so-called ePub files directly. There is already some support for this container format. If you have enabled XHTML export, you can produce an ePub archive afterwards with:

```
mtxrun --script epub yourfile
```

For testing the results, open source programs like `calibre` are quite useful. It will probably take a while to figure out to what extent we need to support formats like ePub, if only because such formats are adapted on a regular basis.

7 The future

It is hard to predict the future. I can imagine that given the user interface that has evolved over ages paper books will not disappear soon. Probably there will be a distinction between read-once and throwaway books and those that you carry with you your whole life as visible proof of that life. I can also imagine that (if only for environmental reasons) ebooks (hopefully with stable devices) will dominate. In that case traditional bookshops will disappear and with them the need for publishers that supply them. Self-publishing will then be most interesting for authors and maybe some of them (or their helpful friends) will be charmed by TEX and tinkering with the layout using the macro language. I can also imagine that at some point new media (and I don't consider an ebook a new medium) will dominate. And how about science fiction becoming true: downloading stories and information directly into our brains.

It reminds me of something I need to do some day soon: get rid of old journals that I planned to read but never will. I would love to keep them electronically but it is quite unlikely that they are available and if so, it's unlikely that I want to pay for them again. This is typically an area where I'd consider using an ebook device, even if it's suboptimal. On the other hand, I don't consider dropping my newspaper subscription yet as I don't see a replacement for the regular coffeestop at the table where it sits and where we discuss the latest news.

The nice thing about an analogue camera is that the image carrier has been standardized and you can buy batteries everywhere. Compare this with their digital cousins: all have different batteries, there are all kinds of memory cards, and only recently has some standardization in lenses shown up. There is a wide range of resolutions and aspect ratios. Other examples of standardization are nuts and bolts used in cars, although it took some time for the metric system to catch on. Books have different dimensions but it's not hard to deal with that property. Where desktop hardware is rather uniform everything portable is different. For some brands you need a special toolkit with every new device. Batteries cannot be exchanged and there are quite some data carriers. On the other hand, we're dealing with software and if we want we can support data formats forever. The Microsoft operating systems have demonstrated that programs written years ago can still run on updates. In addition Linux demonstrates that users can take and keep control and create an independence from vendors. So, given that we can still read document sources and given that they are well structured, we can create history-proof solutions. I don't expect that the traditional publishers will play a big role in this if only because of their short term agendas and because changing ownerships works against long term views. And programs like TEX have already demonstrated having a long life span, although it must be said that in today's rapid upgrade cycles it takes some courage to stay with it and its descendants. But downward compatibility is high on the agenda of its users and user groups which is good in the perspective of discussing stable ebooks.

Let's finish with an observation. Books often make a nice (birthday) present and finding one that suits is part of the gift. Currently a visible book has some advantages: when unwrapped it can be looked at and passed around. It also can be a topic of discussion and it has a visible personal touch. I'm not so sure if vouchers for an ebook have the same properties. It probably feels a bit like giving synthetic flowers. I don't know what percentage of books is given as presents but this aspect cannot be neglected. Anyway, I wonder when I will buy my first ebook and for who. Before that happens I'll probably have generated lots of them.

◇ Hans Hagen
http://pragma-ade.com

[3] Some more advanced MathML output will be available when the matrix-related core commands have been upgraded to MkIV and extended to suit today's needs.