Do we need a ‘Cork’ math font encoding?*

Abstract
The city of Cork has become widely known in the TEX community, ever since it gave name to an encoding developed at the European TEX conference of 1990. The ‘Cork’ encoding, as it became known, was the first example of an 8-bit text font encoding that appeared after the release of TEX 3.0, and was later followed by a number of other encodings based on similar design principles.

As of today, the ‘Cork’ encoding represents only one out of several possible choices of 8-bit subsets from a much larger repertoire of glyphs provided in fonts such as Latin Modern or TEX Gyre. Moreover, recent developments of new TEX engines are making it possible to take advantage of OpenType font technology directly, largely eliminating the need for 8-bit font encodings altogether.

During the entire time since 1990 math fonts have always been lagging behind the developments in text fonts. While the need for new math font encodings was recognized early on and while several encoding proposals have been discussed, none of them ever reached production quality or became widely used.

In this paper, we review the situation of math fonts as of 2008, especially in view of recent developments of Unicode and OpenType math fonts such as the STIX fonts or Cambria Math. In particular, we try to answer the question whether a ‘Cork’ math font encoding is still needed or whether Unicode and OpenType might eliminate the need for TEX-specific math font encodings.

History and development of text fonts

The ‘Cork’ encoding

When the 5th European TEX conference was held in Cork in the summer of 1990, the TEX community was undergoing a major transition phase. TEX 3.0 had just been released that year, making it possible to switch from 7-bit to 8-bit font encodings and to support hyphenation for multiple languages.

Since the ability to properly typeset and hyphenate accented languages strongly depended on overcoming the previous limitations, European TEX users wanted to take advantage of the new features and started to work on new font encodings [1, 2, 3]. As a result, they came up with an encoding that became widely known as the ‘Cork’ encoding, named after the site of the conference [4].

The informal encoding name ‘Cork’ stayed in use for many years, even after LaTEX 2$_\varepsilon$ and NFSS2 introduced a system of formal encoding names in 1993–94, assigning `OT`n for 7-bit old text encodings, `T`n for 8-bit standard text encodings, and `L`n for local or non-standard encodings [5]. The ‘Cork’ encoding was the first example of a standard 8-bit text font and thus became the `T1` encoding.

While the ‘Cork’ encoding was certainly an important achievement, it also introduced some novel features that may have seemed like a good idea at that time but would be seen as shortcomings or problems from today’s point of view, after nearly two decades of experience with font encodings.

In retrospect, the ‘Cork’ encoding represents a typical example of the TEX-specific way of doing things of the early 1990s without much regard for standards or technologies outside the TEX world.

Instead of following established standards, such as using ISO Latin 1 or 2 or some extended versions for Western and Eastern European languages, the ‘Cork’ encoding tried to support as many languages as possible in a single font encoding, filling the 8-bit font table to the limit with accented characters at the expense of symbols. Since there was no more room left in the font table, typesetting symbols at first had to be taken from the old 7-bit fonts, until a supplementary text symbol `TS1` encoding [6] was introduced in 1995 to fill the gap.

When it came to implementing the `T1` and `TS1` encodings for PostScript fonts, it turned out that the encodings were designed without taking into account the range of glyphs commonly available in standard PostScript fonts.

Both font encodings could only be partially implemented with glyphs from the real font, while the remaining slots either had to be faked with virtual fonts or remain unavailable. At the same time, none of the encodings provided access to the full set of available glyphs from the real font.

*First published in *TUGboat* **29**:3 (2008), pp.426–434. Reprinted with permission.

Alternatives to the 'Cork' encoding

As an alternative to using the T1 and TS1 encodings for PostScript fonts, the TeXnANSI or LY1 encoding was proposed [7], which was designed to provide access to the full range of commonly available symbols (similar to the TeXBase1 encoding), but also matched the layout of the OT1 encoding in the lower half, so that it could be used as drop-in replacement without any need for virtual fonts.

In addition to that, a number of non-standard encodings have come into use as local alternatives to the 'Cork' encoding, such as the Polish QX, the Czech CS, and the Lithuanian L7X encoding, each of them trying to provide better solutions for the needs of specific languages.

In summary, the 'Cork' encoding as the first example of an 8-bit text encoding (T1) was not only followed by additional encodings based on the same design principles for other languages (T*n*), but also supplemented by a text symbol encoding (TS1) and complemented by a variety of local or non-standard encodings (LY1, QX, CS, etc.).

As became clear over time, the original goal of the 'Cork' encoding of providing a single standard encoding for as many languages as possible couldn't possibly be achieved within the limits of 8-bit fonts, simply because there are far too many languages and symbols to consider, even when limiting the scope to Latin and possibly Cyrillic or Greek.

Recent developments of text fonts

Unicode support in new TEX fonts

It was only in recent years that the development of the Latin Modern [8, 9, 10] and TEX Gyre fonts [11, 12] has provided a consistent implementation for all the many choices of encodings.

As of today, the 'Cork' encoding represents only one out of several possible 8-bit subsets taken from a much larger repertoire of glyphs. The full set of glyphs, however, can be accessed only when moving beyond the limits of 8-bit fonts towards Unicode and OpenType font technology.

Unicode support in new TEX engines

As we are approaching the TUG 2008 conference at Cork, the TEX community is again undergoing a major transition phase. While TEX itself remains frozen and stable, a number of important developments have been going on in recent years.

Starting with the development of PDFTEX since the late 1990s the use of PDF output and scalable PostScript or TrueType fonts has largely replaced the use of DVI output and bitmap PK fonts.

Followed by the ongoing development of XƎTEX and LuaTEX in recent years the use of Unicode and OpenType font technology is also starting to replace the use of 8-bit font encodings as well as traditional PostScript or TrueType font formats.

Putting everything together, the development of new fonts and new TEX engines in recent years has enabled the TEX community to catch up with developments of font technology in the publishing industry and to prepare for the future.

The only thing still missing (besides finishing the ongoing development work) is the development of support for Unicode math in the new TEX engines and the development of OpenType math fonts for Latin Modern and TEX Gyre.

History and development of math fonts

When TEX was first developed in 1977–78, the 7-bit font encodings for text fonts and math fonts were developed simultaneously, since both of them were needed for typesetting mathematical textbooks like *The Art of Computer Programming*.

When TEX 3.0 made it possible to switch from 7-bit to 8-bit font encodings, it was the text fonts driving these new developments while the math fonts remained largely unchanged.

As a result, the development of math fonts has been lagging behind the corresponding text fonts for nearly two decades now, ever since the development of the 'Cork' encoding started in 1990.

In principle, a general need for new math fonts was recognized early on: When the first implementations of 'Cork' encoded text fonts became available, it was soon discovered that the new 8-bit text fonts couldn't fully replace the old 7-bit text fonts without resolving the inter-dependencies between text and math fonts. In practice, however, nothing much happened since there was no pressing need.

The 'Aston' proposal

The first bit of progress was made in the summer of 1993, when the LaTEX3 Project and some TEX users group sponsored a research student to work on math font encodings for a few months.

As a result, a proposal for the general layout of new 8-bit math font encodings was developed and presented at TUG 1993 at Aston University [13]. Unlike the 'Cork' encoding, which became widely known, this 'Aston' proposal was known only to some insiders and went largely unnoticed.

After only a few months of activity in 1993 the project mailing list went silent and nothing further happened for several years, even after a detailed report was published as a LaTEX3 Project Report [14].

The 'newmath' prototype

The next bit of progress was made in 1997–98, when the ideas of the 'Aston' proposal were taken up again and work on an implementation was started.

This time, instead of just discussing ideas or preparing research documents, the project focussed on developing a prototype implementation of new math fonts for several font families using a mixture of MetaFont and `fontinst` work [15].

When the results of the project were presented at the EuroTEX 1998 conference [16], the project was making good progress, although the results were still very preliminary and far from ready for production.

Unfortunately, the project then came to a halt soon after the conference when other activities came to the forefront and changed the scope and direction of the project [17, 18].

Before the conference, the goal of the project had been to develop a set of 8-bit math font encodings for use with traditional TEX engines (within the constraints of 16 families of 256 glyphs) and also to provide some example implementations by means of reencoding and enhancing existing font sets.

After the conference, that goal was set aside and put on hold for an indefinite time by the efforts to bring math into Unicode.

Recent developments of math fonts

Unicode math and the stix fonts

While the efforts to bring math into Unicode were certainly very important, they also brought along a lot of baggage in the form of a very large number of additional symbols, making it much more work to provide a reasonably complete implementation and nearly impossible to encode all those symbols within the constraints of traditional TEX engines.

In the end, the Unicode math efforts continued over several years until the symbols were accepted [19, 20] and several more years until an implementation of a Unicode math font was commissioned [21] by a consortium of scientific and technical publishers, known as the STIX Project.

When the first beta-test release of the so-called STIX fonts [22] finally became available in late 2007, nearly a decade had passed without making progress on math font encodings for TEX.

While the STIX fonts provide all the building blocks of Unicode math symbols, they are still lacking TEX support and may yet have to be repackaged in a different way to turn them into a usable font for use with TEX or other systems.

Despite the progress on providing the Unicode math symbols, the question of how to encode all the many Unicode math symbols in a set of 8-bit font encodings for use with traditional TEX engines still remains unresolved. Most likely, only a subset of the most commonly used symbols could be made available in a set of 8-bit fonts, whereas the full range of symbols would be available only when moving to Unicode and OpenType font technology.

OpenType math in ms Office 2007

While the TEX community and the consortium of scientific publishers were patiently awaiting the release of the STIX fonts before reconsidering the topic of math font encodings, outside developments have continued to move on. In particular, Microsoft has moved ahead and has implemented its own support for Unicode math in Office 2007.

They did so by adding support for math typesetting in OpenType font technology [23, 24] and by commissioning the design of the Cambria Math font as an implementation of an OpenType math font [25, 26, 27]. In addition, they have also adopted an input language called 'linear math' [28], which is strongly based on TEX concepts.

While OpenType math is officially still considered experimental and not yet part of the OpenType specification [29], it is already a *de facto* standard, not only because it has been deployed to millions of installations of Office 2007, but also because it has already been adopted by other projects, such as the FontForge font editor [30] and independent font designs such as Asana Math [31].

In addition, the next release of the STIX fonts scheduled for the summer of 2008 is also expected to include support for OpenType math.

OpenType math in new TEX engines

At the time of writing, current development versions of XƎTEX have added some (limited) support for OpenType math, so it is already possible to use fonts such as Cambria Math in XƎTEX [32], and this OpenType math support will soon become available to the TEX community at large with the upcoming release of TEX Live 2008.

Most likely, LuaTEX will also be adding support for OpenType math eventually, so OpenType math is likely to become a *de facto* standard in the TEX world as well, much as we have adopted other outside developments in the past.

OpenType math for new TEX fonts?

Given these developments, the question posed in the title of this paper about the need for new math font encodings may soon become a non-issue.

If we decide to adopt Unicode and OpenType math font technology in new TEX engines and new fonts, the real question is no longer how to design the layout of encoding tables but rather how to deal with the technology of OpenType math fonts, as we will discuss in the following sections.

Future developments in math fonts

Some background on OpenType math

The OpenType font format was developed jointly by Microsoft and Adobe, based on concepts adopted from the earlier TrueType and PostScript formats. The overall structure of OpenType fonts shares the extensible table structure of TrueType fonts, adding support for different flavors of glyph descriptions in either PostScript CFF or TrueType format. (An extensive documentation of the OpenType format and its features as well as many other important font formats can be found in [33].)

One of the most interesting points about OpenType is the support for 'advanced' typographic features, supporting a considerable amount of intelligence in the font, enabling complex manipulations of glyph positioning or glyph substitutions. At the user level, many of these 'advanced' typographic features can be controlled selectively by the activation of so-called OpenType feature tags.

Despite its name, the OpenType font format is not really open and remains a vendor-controlled specification, much like the previous TrueType and PostScript font formats developed by these vendors. The official OpenType specification is published on a Microsoft web site at [29], but that version may not necessarily reflect the latest developments.

In the case of OpenType math, Microsoft has used its powers as one of the vendors controlling the specification to implement an extension of the OpenType format and declare it as 'experimental' until they see fit to release it. Fortunately, Microsoft was smart enough to borrow from the best examples of math typesetting technology when they designed OpenType math, so they chose TEX as a model for many of the concepts of OpenType math.

The details of OpenType math

The OpenType* MATH *table. One of the most distinctive features of an OpenType math font is the presence of a MATH table. This table contains a number of global font metric parameters, much like the `\fontdimen` parameters of math fonts in TEX described in Appendix G of *The TEXbook*.

In a traditional TEX setup these parameters are essential for typesetting math, controlling various aspects such as the spacing of elements such as big operators, fractions, and indices [34, 35].

In an OpenType font the parameters of the MATH table have a similar role for typesetting math. From what is known, Microsoft apparently consulted with Don Knuth about the design of this table, so the result is not only similar to TEX, but even goes beyond TEX by adding new parameters for cases where hard-wired defaults are applied in TEX.

In the XƎTEX implementation the parameters of the OpenType MATH table are mapped internally to TEX's `\fontdimen` parameters. In most cases this mapping is quite obvious and straight-forward, but unfortunately there are also a few exceptions where some parameters in TEX do not have a direct correspondence in OpenType. It is not clear, however, whether these omissions are just an oversight or a deliberate design decision in case a parameter was deemed irrelevant or unnecessary.

Support for OpenType math in XƎTEX still remains somewhat limited for precisely this reason; until the mapping problems are resolved, XƎTEX has to rely on workarounds to extract the necessary parameters from the OpenType MATH table.

At the time of writing, the extra parameters introduced by OpenType generalizing the concepts of TEX have been silently ignored. It is conceivable, however, that future extensions of new TEX engines might eventually start to use these parameters in the math typesetting algorithms as well.

In the end, whatever technology is used to typeset OpenType math, it remains the responsibility of the font designer to set up the values of all the many parameters affecting the quality of math typesetting. Unfortunately, for a non-technical designer such a task feels like a burden, which is better left to a technical person as a font implementor.

For best results, it is essential to develop a good understanding of the significance of the parameters and how they affect the quality of math typesetting. In [35] we have presented a method for setting up the values of metric parameters of math fonts in TEX. For OpenType math fonts, we would obviously have to reconsider this procedure.

Font metrics of math fonts. Besides storing the global font metric parameters, the OpenType MATH table is also used to store additional glyph-specific information such as italic corrections or kern pairs, as well as information related to the placement of math accents, superscripts and subscripts.

In a traditional TEX setup the font metrics of math fonts have rather peculiar properties, because much of the glyph-specific information is encoded or hidden by overloading existing fields in the TFM metrics in an unusual or non-intuitive way [36].

For example, the width in the TFM metrics is not the real width of the glyph. Instead, it is used to indicate the position where to attach the subscript. Similarly, the italic correction is used to indicate the offset between subscript and superscript.

As another example, fake kern pairs involving a skewchar are used to indicate how much the visual center of the glyphs is skewed in order to determine the position where to attach a math accent.

In OpenType math fonts all such peculiarities will become obsolete, as the MATH table provides data structures to store all the glyph-specific metric information in a much better way. In the case of indices, OpenType math has extended the concepts of TEX by defining 'cut-ins' at the corners on both sides of a glyph and not just to the right.

Unfortunately, while the conceptual clarity of OpenType math may be very welcome in principle, it may cause an additional burden on font designers developing OpenType math fonts based on traditional TEX fonts (such as the Latin Modern fonts) and trying to maintain metric compatibility.

In such cases it may be necessary to examine the metrics of each glyph and to translate the original metrics into appropriate OpenType metrics.

Font encoding and organization. The encoding of OpenType fonts is essentially defined by Unicode code points. Most likely, a typical OpenType math font will include only a subset of Unicode limited to the relevant ranges of math symbols and alphabets, while the corresponding text font may contain a bigger range of scripts.

In a traditional TEX setup the math setup consists of a series of 8-bit fonts organized into families. Typically, each font will contain one set of alphabets in a particular style and a selection of symbols filling the remaining slots.

In a Unicode setup the math setup will consist of only one big OpenType font, containing all the math symbols and operators in the relevant Unicode slots, as well as all the many styles of math alphabets assigned to slots starting at `U+1D400`.

As a result, there will be several important conceptual implications to consider in the design and implementation of OpenType math fonts, such as how to handle font switches of math alphabets, how to include the various sizes of big operators, delimiters, or radicals, or how to include the optical sizes of superscripts and subscripts.

Handling of math alphabets. In a traditional TEX setup the letters of the Latin and Greek alphabets are subject to font switches between the various math families, usually containing a different style in each family (roman, italic, script, etc.).

In a Unicode setup each style of math alphabets has a different range of slots assigned to it, since each style is assumed to convey a different meaning.

When dealing with direct Unicode input, this might not be a problem, but when dealing with traditional TEX input, quite a lot of setup may be needed at the macro level to ensure that input such as `\mathrm{a}` or `\mathit{a}` or `\mathbf{a}` will be translated to the appropriate Unicode slots.

An additional complication arises because the Unicode code points assigned to the math alphabets are non-contiguous for historical reasons [37]. While most of the alphabetic letters are taken from one big block starting at `U+1D400`, a few letters which were part of Unicode already before the introduction of Unicode math have to be taken from another block starting at `U+2100`.

An example implementation of a LaTEX macro package for XƎTEX to support OpenType math is already available [32], and it shows how much setup is needed just to handle math alphabets. Fortunately, such a setup will be needed only once and will be applicable for all Unicode math fonts, quite unlike the case of traditional TEX fonts where each set of math fonts requires its own macro package.

Handling of size variants. Ever since the days of DVI files and PK fonts, TEX users have been accustomed to thinking of font encodings in terms of numeric slots in an encoding table, usually assuming a 1:1 mapping between code points and glyphs.

However, there have always been exceptions to this rule, most notably in the case of a math extension font, where special TFM features were used to set up a linked list from one code point to a series of next-larger glyph variants representing different sizes of operators, delimiters, radicals, or accents, optionally followed by an extensible version.

In a traditional TEX font each glyph variant has a slot by itself in the font encoding, even if it was addressed only indirectly.

In an OpenType font, however, the font encoding is determined by Unicode code points, so the additional glyph variants representing different sizes cannot be addressed directly by Unicode code points and have to remain unencoded, potentially mapped to the Unicode private use area, if needed.

While the conceptual ideas of vertical and horizontal variants and constructions in the OpenType MATH table are very similar to the concepts of charlists and extensible recipes in TEX font metrics, it is interesting to note that OpenType has generalized these concepts a little bit.

While TEX supports extensible recipes only in a vertical context of big delimiters, OpenType also supports horizontal extensible constructions, so it would be possible to define an extensible overbrace or underbrace in the font, rather than at the macro level using straight line segments for the extensible parts. In addition, the same concept could also be applied to arbitrarily long arrows.

Optical sizes for scripts. In a traditional TEX setup math fonts are organized into families, each of them consisting of three fonts loaded at different design sizes representing text style and first and second level script style.

If a math font provides optical design sizes, such as in the case of traditional MetaFont fonts, these fonts are typically loaded at sizes of 10 pt, 7 pt, 5 pt, each of them having different proportions adjusted for improved readability at smaller sizes.

If a math font doesn't provide optical sizes, such as in the case of typical PostScript fonts, scaled-down versions of the 10 pt design size will have to make do, but in such cases it may be necessary to use bigger sizes of first and second level scripts, such as 10 pt, 7.6 pt, 6 pt, since the font may otherwise become too unreadable at such small sizes.

In OpenType math the concept of optical sizes from TEX and MetaFont has been adopted as well, but it is implemented in a different way, typical for OpenType fonts. Instead of loading multiple fonts at different sizes, OpenType math fonts incorporate the multiple design variants in the same font and activate them by a standard OpenType substitution mechanism using a feature tag `ssty=0` and `ssty=1`, not much different from the standard substitutions for small caps or oldstyle figures in text fonts.

It is important to note that the optical design variants intended for use in first and second level scripts, using proportions adjusted for smaller sizes, are nevertheless provided at the basic design size and subsequently scaled down using a scaling factor defined in the OpenType MATH table.

If an OpenType math font lacks optical design variants for script sizes and does not support the `ssty` feature tag, a scaled-down version of the basic design size will be used automatically. The same will also apply to non-alphabetic symbols.

Use of OpenType feature tags. Besides using OpenType feature tags for specific purposes in math fonts, most professional OpenType text fonts also use feature tags for other purposes, such as for selecting small caps or switching between oldstyle and lining figures. Some OpenType fonts may provide a rich set of features, such as a number of stylistic variants, initial and final forms, or optical sizes.

Ultimately, it remains to be seen how the use of OpenType feature tags will influence the organization of OpenType fonts for TEX, such as Latin Modern or TEX Gyre, not just concerning new math fonts, but also existing text fonts.

So far, the Latin Modern fonts have very closely followed the model of the Computer Modern fonts, providing separate fonts for each design size and each font shape or variant.

While it might well be possible to eliminate some variants by making extensive use of OpenType feature tags, such as by embedding small caps into the roman fonts, implementing such a step would imply an important conceptual change and might cause unforeseen problems.

Incorporating multiple design sizes into a single font might have similar implications, but the effects might be less critical if they are limited to the well-controlled environment of math typesetting.

In the TEX Gyre fonts the situation is somewhat simpler, because these fonts are currently limited to the basic roman and italic fonts and do not have small caps variants or optical sizes.

Incorporating a potential addition of small caps in TEX Gyre fonts by means of OpenType feature tags might well be possible without causing any incompatible changes. Similarly, incorporating some expanded design variants with adjusted proportions for use in script sizes would also be conceivable when designing TEX Gyre math fonts.

The impact of OpenType math

As we have seen in the previous sections, OpenType math fonts provide a way of embedding all the relevant font-specific and glyph-specific information needed for high-quality math typesetting.

In many aspects, the concepts of OpenType math are very similar to TEX or go beyond TEX. However, the implementation of these concepts in OpenType fonts will be different in most cases.

Given the adoption of OpenType math as a *de facto* standard and its likelihood of becoming an official standard eventually, OpenType math seems to be the best choice for future developments of new math fonts for use with new TEX engines.

While XƎTEX has already started to support OpenType math and LuaTEX is very likely to follow, adopting OpenType for the design of math fonts for Latin Modern or TEX Gyre will take more time and will require developing a deeper understanding of the concepts and data structures.

Most importantly, however, it will also require rethinking many traditional assumptions about the way fonts are organized.

Thus, while the topic of font encodings of math fonts may ultimately become a non-issue, the topic of font technology will certainly remain important.

The challenges of OpenType math

Developing a math font has never been an easy job, so attempting to develop a full-featured OpenType math font for Latin Modern or TEX Gyre certainly presents a major challenge to font designers or font implementors for a number reasons.

First, such a math font will be really large, even in comparison with text fonts, which already cover a large range of Unicode. (In the example of the Cambria Math font, the math font is reported to have more than 2900 glyphs compared to nearly 1000 glyphs in the Cambria text font.) It will have to extend across multiple 16-bit planes to account for the slots of the math alphabets starting at `U+1D400`, and it will also require a considerable number of unencoded glyphs to account for the size variants of extensible glyphs and the optical variants of math alphabets.

Besides the size of the font, such a project will also present many technical challenges in dealing with the technology of OpenType math fonts.

While setting up the font-specific parameters of the OpenType MATH table is comparable to setting up the `\fontdimen` parameters of TEX's math fonts, setting up the glyph-specific information will require detailed attention to each glyph as well as extensive testing and fine-tuning to achieve optimal placement of math accents and indices.

Finally, there will be the question of assembling the many diverse elements that have to be integrated in a comprehensive OpenType math font. So far, the various styles of math alphabets and the various optical sizes of these alphabets have been designed as individual fonts, but in OpenType all of them have to be combined in a single font. Moreover, the optical sizes will have to be set up as substitutions triggered by OpenType feature tags.

Summary and conclusions

In this paper we have reviewed the work on math font encodings since 1990 and the current situation of math fonts as of 2008, especially in view of recent developments in Unicode and OpenType font technology. In particular, we have looked in detail at the features of OpenType math in comparison to the well-known features of TEX's math fonts.

While OpenType math font technology looks very promising and seems to be the best choice for future developments of math fonts, it also presents many challenges that will have to be met.

While support for OpenType math in new TEX engines has already started to appear, the development of math fonts for Latin Modern or TEX Gyre using this font technology will not be easy and will take considerable time.

In the past, the TEX conference in Cork in 1990 was the starting point for major developments in text fonts, which have ultimately led to the adoption of Unicode and OpenType font technology.

Hopefully, the TEX conference at Cork in 2008 might become the starting point for major developments of math fonts in a similar way, except that this time there will be no more need for a new encoding that could be named after the site of the conference.

Acknowledgements

The author wishes to acknowledge feedback, suggestions, and corrections from some of the developers of projects discussed in this review.

A preprint of this paper has been circulated on the Unicode math mailing list hosted at Google Groups [38] and future discussions on the topics of this paper are invited to be directed here.

References

[1] Yannis Haralambous: TEX and Latin alphabet languages. *TUGboat*, 10(3):342–345, 1989. `http://www.tug.org/TUGboat/Articles/tb10-3/tb25hara-latin.pdf`

[2] Nelson Beebe: Character set encoding. *TUGboat*, 11(2):171–175, 1990. `http://www.tug.org/TUGboat/Articles/tb11-2/tb28beebe.pdf`

[3] Janusz S. Bień: On standards for CM font extensions. *TUGboat*, 11(2):175–183, 1990. `http://www.tug.org/TUGboat/Articles/tb11-2/tb28bien.pdf`

[4] Michael Ferguson: Report on multilingual activities. *TUGboat*, 11(4):514–516, 1990. `http://www.tug.org/TUGboat/Articles/tb11-4/tb30ferguson.pdf`

[5] Frank Mittelbach, Robin Fairbairns, Werner Lemberg: LaTEX font encodings, 2006. `http://www.ctan.org/tex-archive/macros/latex/doc/encguide.pdf`

[6] Jörg Knappen: The release 1.2 of the Cork encoded DC fonts and text companion fonts. *TUGboat*, 16(4):381–387, 1995. Reprint from the Proceedings of the 9th European TEX Conference 1995, Arnhem, The Netherlands. `http://www.tug.org/TUGboat/Articles/tb16-4/tb49knap.pdf`

[7] Berthold K. P. Horn: The European Modern fonts. *TUGboat*, 19(1):62–63, 1998
`http://www.tug.org/TUGboat/Articles/tb19-1/tb58horn.pdf`

[8] Bogusław Jackowski, Janusz M. Nowacki: Latin Modern: Enhancing Computer Modern with accents, accents, accents. *TUGboat*, 24(1):64–74, 2003. Proceedings of the TUG 2003 Conference, Hawaii, USA.
`http://www.tug.org/TUGboat/Articles/tb24-1/jackowski.pdf`

[9] Bogusław Jackowski, Janusz M. Nowacki: Latin Modern: How less means more. *TUGboat*, 27(0):171–178, 2006 Proceedings of the 15th European TEX Conference 2005, Pont-à-Mousson, France.
`http://www.tug.org/TUGboat/Articles/tb27-0/jackowski.pdf`

[10] Will Robertson: An exploration of the Latin Modern fonts. *TUGboat*, 28(2):177-180, 2007.
`http://www.tug.org/TUGboat/Articles/tb28-2/tb89robertson.pdf`

[11] Hans Hagen, Jerzy B. Ludwichowski, Volker RW Schaa: The new font project: TEX Gyre. *TUGboat*, 27(2):250–253, 2006. Proceedings of the TUG 2006 Conference, Marrakesh, Morocco.
`http://www.tug.org/TUGboat/Articles/tb27-2/tb87hagen-gyre.pdf`

[12] Jerzy B. Ludwichowski, Bogusław Jackowski, Janusz M. Nowacki: Five years after: Report on international TEX font projects. *TUGboat*, 29(1):25–26, 2008. Proceedings of the 17th European TEX Conference 2007, Bachotek, Poland.
`https://www.tug.org/TUGboat/Articles/tb29-1/tb91ludwichowski-fonts.pdf`

[13] Alan Jeffrey: Math font encodings: A workshop summary. *TUGboat*, 14(3):293–295, 1993. Proceedings of the TUG 1993 Conference, Aston University, Birmingham, UK.
`http://www.tug.org/TUGboat/Articles/tb14-3/tb40mathenc.pdf`

[14] Justin Ziegler: Technical report on math font encodings. LaTEX3 Project Report, 1993.
`http://www.ctan.org/tex-archive/info/ltx3pub/processed/l3d007.pdf`

[15] Math Font Group (MFG) web site, archives, papers, and mailing list.
`http://www.tug.org/twg/mfg/`
`http://www.tug.org/twg/mfg/archive/`
`http://www.tug.org/twg/mfg/papers/`
`http://www.tug.org/mailman/listinfo/math-font-discuss`

[16] Matthias Clasen, Ulrik Vieth: Towards a new Math Font Encoding for AllTEX. *Cahiers GUTenberg*, 28–29:94–121, 1998. Proceedings of the 10th European TEX Conference 1998, St. Malo, France.
`http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/28-29-clasen.pdf`

[17] Ulrik Vieth *et al.*: Summary of math font-related activities at EuroTEX 1998. *MAPS*, 20:243–246, 1998.
`http://www.ntg.nl/maps/20/36.pdf`

[18] Ulrik Vieth: What is the status of new math font encodings? Posting to mailing list, 2007.
`http:/www./tug.org/pipermail/math-font-discuss/2007-May/000068.html`

[19] Barbara Beeton, Asmus Freytag, Murray Sargent III: Unicode Support for Mathematics. Unicode Technical Report UTR#25. 2001.
`http://www.unicode.org/reports/tr25/`

[20] Barbara Beeton: Unicode and math, a combination whose time has come–Finally! *TUGboat*, 21(3):174–185, 2000. Proceedings of the TUG 2000 Conference, Oxford, UK.
`http://www.tug.org/TUGboat/Articles/tb21-3/tb68beet.pdf`

[21] Barbara Beeton: The STIX Project–From Unicode to fonts. *TUGboat*, 28(3):299–304, 2007. Proceedings of the TUG 2007 Conference, San Diego, California, USA.
`http://www.tug.org/TUGboat/Articles/tb28-3/tb90beet.pdf`

[22] STIX Fonts Project: Web Site and Frequently Asked Questions.
`http://www.stixfonts.org/`
`http://www.stixfonts.org/STIXfaq.html`

[23] Murray Sargent III: Math in Office Blog.
`http://blogs.msdn.com/murrays/default.aspx`

[24] Murray Sargent III: High-quality editing and display of mathematical text in Office 2007.
`http://blogs.msdn.com/murrays/archive/2006/09/13/752206.aspx`

[25] Tiro Typeworks: Cambria Math Specimen.
`http://www.tiro.nu/Articles/Cambria/Cambria_Math_Basic_Spec_V1.pdf`

[26] John Hudson, Ross Mills: Mathematical Typesetting: Mathematical and scientific typesetting solutions from Microsoft. Promotional Booklet, Microsoft, 2006.
`http://www.tiro.com/projects/`

[27] Daniel Rhatigan: Three typefaces for mathematics. The development of Times 4-line Mathematics, AMS Euler, and Cambria Math. Dissertation for the MA in typeface design,

University of Reading, 2007.
http://www.typeculture.com/academic_resource/articles_essays/pdfs/tc_article_47.pdf

[28] Murray Sargent III: Unicode Nearly Plain Text Encodings of Mathematics. Unicode Technical Note UTN#28, 2006.
http://www.unicode.org/notes/tn28/

[29] Microsoft Typography: OpenType specification version 1.5.
http://www.microsoft.com/typography/otspec/

[30] George Williams: FontForge. Math typesetting information.
http://fontforge.sourceforge.net/math.html

[31] Apostolos Syropoulos: Asana Math.
www.ctan.org/tex-archive/fonts/Asana-Math/

[32] Will Robertson: Experimental Unicode math typesetting: The unicode-math package.
http://github.com/wspr/unicode-math/tree/master

[33] Yannis Haralambous: Fonts and Encodings. O'Reilly Media, 2007. ISBN 0-596-10242-9
http://oreilly.com/catalog/9780596102425/

[34] Bogusław Jackowski: Appendix G Illuminated. *TUGboat*, 27(1):83–90, 2006. Proceedings of the 16th European TEX Conference 2006, Debrecen, Hungary.
http://www.tug.org/TUGboat/Articles/tb27-1/tb86jackowski.pdf

[35] Ulrik Vieth: Understanding the æsthetics of math typesetting. *Biuletyn* GUST, 5–12, 2008. Proceedings of the 16th BachoTEX Conference 2008, Bachotek, Poland.
http://www.gust.org.pl/projects/e-foundry/math-support/vieth2008.pdf

[36] Ulrik Vieth: Math Typesetting in TEX: The Good, the Bad, the Ugly. *MAPS*, 26:207–216, 2001. Proceedings of the 12th European TEX Conference 2001, Kerkrade, Netherlands.
http://www.ntg.nl/maps/26/27.pdf

[37] Unicode Consortium: Code Charts for Symbols and Punctuation.
http://www.unicode.org/charts/symbols.html

[38] Google Groups: Unicode math for TEX.
http://groups.google.com/group/unimath

Ulrik Vieth
Vaihinger Straße 69
70567 Stuttgart
Germany
ulrik dot vieth (at) arcor dot de