An OpenType font with built-in TEX syntax highlighting

K.V. Rajeesh, C.V. Radhakrishnan

Abstract

In code editors, typesetting applications, and on the Web, syntax-highlighting is achieved via specialized 'grammar' files or JavaScript+CSS libraries. Using the colour table in OpenType fonts, we present development of a monospace font with basic built-in syntax highlighting for TEX and LATEX documents.

1 The necessity

Syntax highlighting is a useful — or distracting for some — functionality offered, usually by default, in programming editors, IDEs, typesetting applications and on web pages. Each requires a different approach, e.g.:

- Emacs: language mode for syntax coloring.
- Vim: syntax files loaded on-demand.
- Kate/Kwrite/Qt, etc.: theme files.
- Web: JavaScript frameworks like `highlight.js`, `Prism`, etc.
- TEX and friends: packages such as `minted` [6], `listings` [3], etc. In particular, `minted` requires shell escaping (mitigated in the latest release), which is a security risk in production systems.

One of the use cases is highlighting TEX code snippets (both inline and block) on instructional wiki pages for authors and editors. The general solution for Mediawiki is to use a suitable PHP extension, but those are performance intensive, slowing down page loads. A font that highlights TEX syntax is a lightweight alternate solution.

Inspired by prior art (see Section 2.1), the authors developed a font which highlights TEX and LATEX syntax automatically without requiring any extensions or JavaScript. The end result is a freely available typeface shown in Fig. 1.

2 Implementation details

A font today is a dynamic and complex piece of software. It contains instructions to draw and rasterize glyph shapes from vector data and much metadata about the contained glyphs: their names, indices, encoding, etc., font family, styles, ascent/descent values, and much more. The most widely used font format, viz. OpenType [5], has extensible specifications to add new functionality to fonts in backward-compatible ways (i.e. new functionality added to a font may not be supported by an existing software application, but will still continue to work as before.)

```
\documentclass[12pt]{article}
\makeatletter
\def\my@emphstyle#1{\csname my@style@#1\endcsname}
\makeatother

\begin{document}
\section{Introduction}
An OpenType COLRv1 font with \TeX syntax highlighting.
With math: $a+b^c_d+c^2-\eta$
\par
And environments
\begin{thm}[Pythagoras' Theorem]
Handle optional parameters \citet*[e.g.][here]{key_one}.
\end{thm}
Some \textit{text here}.
Some %comment $\abcd+efg$ on this line
\end{document}
```

Figure 1: OpenType COLRv1 font with built-in TEX syntax highlighting, rendered by a web browser.

Technically, each functionality is implemented as a 'table' in the OpenType binary font format.

Two OpenType features of particular interest to our development are programmability and colour.

Programmability OpenType fonts (either TTF or OTF format) have embedded, limited programmability through the GSUB[1] (and GPOS[2], etc.) tables.

The purpose of the GSUB table is 'glyph substitutions'. E.g. a sequence of 'f' and 'i' is 'substituted' to form the 'fi' Latin ligature. Indic scripts require the substitution ability to form consonant cluster-vowel conjuncts to display proper graphemes.

One of the many GSUB features is the `calt` (contextual alternate) feature. It allows for substituting one or more glyphs in sequence with another glyph, 'in the context of' other glyphs — either preceding or following the glyphs to replace. E.g. if `c d` should be substituted with `g` in a sequence of `a b c d e`, it is possible to express this logic via `calt` feature substitution lookups (see Listing 3).

Colour Around 2010, multiple competing colour formats were implemented in OpenType to meet the biggest demand of humanity then: to tweet colourful emojis.

- Apple-backed SBIX format.
- Google-backed CBLC/CBDT format.
- Adobe and Mozilla-backed OT-SVG format.
- Microsoft-backed CPAL/COLR format.

Each format has its own advantages and disadvantages, ranging from rasterizing and rendering

[1] GSUB: glyph substitution. E.g. ligatures are implemented via GSUB. Optional for Latin scripts but mandatory for complex scripts to get proper grapheme representations.

[2] GPOS: glyph positioning. E.g. so-called 'mark positioning' is implemented via GPOS.

doi.org/10.47397/tb/46-2/tb143rajeesh-colorfont-syntax

complexity to flexibility (all formats except OT-SVG are bitmap-based). As a true manifestation of the famous XKCD comic,[3] in 2019 Google proposed a new version of CPAL/COLR format combining the expressiveness of OT-SVG, now known as COLRv1 (the old version is known as COLRv0) that aims to provide flexibility (color blending, transformations, etc.) and performance/renderer simplicity [1]. Luckily, COLRv1 has been widely adopted by major platforms, browsers, and rasterizers (e.g. FreeType) — except for the Safari browser.

The CPAL[4] table is a set of one or more colour palettes, each containing a number of colour records. Each colour record specifies a colour in sRGB colour space using 8-bit BGRA (blue, green, red, alpha) representation.

The COLR[5] table defines colour representation for glyphs, as a graphic composition using 'base' glyphs. COLRv0 allows only for simple composition as a linear sequence of glyphs stacked vertically in bottom-up z-order. COLRv1 additionally supports gradient fills, affine transformations and blending modes; and it supports variable fonts.

2.1 Prior art

Hackers (née tinkerers) have a habit of (ab)using technology for purposes other than its originally intended functionality. Heikki Lotvonen did exactly the same with OpenType `calt` rules and the COLR specifications to develop a font that highlights HTML and CSS snippets [4]. Our approach owes a great deal of inspiration to this work.

2.2 Implementing COLRv1 font and syntax highlighting

Based on the approach described in [4], we chose UFO[6] source format, as it is supported by open source tools such as FontForge [9] and `fontmake` [2].

A condensed and legible open-licensed font 'M PLUS Code Latin 50'[7] is chosen as the base font.

A colour palette of six base colours is chosen to aid readability in both light and dark backgrounds:

- control sequences: red brown
- optional arguments to environments & `\cite`: gray
- comments: gray
- environment names: steel blue

```
<!-- Colour palette -->
<key>
com.github.googlei18n.ufo2ft.colorPalettes
</key>
<array>
  <array>
    <array>
      <!-- red brown, index 0 -->
      <real>0.564706</real>
      <real>0.047059</real>
      <real>0.247059</real>
      <real>1.0</real>
    </array>
    <array>
      <!-- gray, index 1 -->
      <real>0.7</real> <real>0.7</real>
      <real>0.7</real> <real>1.0</real>
    </array>
    ...
  </array>
</array>
```

Listing 1: Excerpt of the colour palette specification for `fontmake`.

- numbers: light purple
- math switch `$`, superscript `^`, subscript `_`: pink
- `@`, macro parameter `#`: pink
- `+`, `-`, `*`, `/` symbols/operators: orange
- delimiters `{}[]|`: orange

2.2.1 Setting colour palette and layers

Glyphs for alphabets `A-Za-z` are displayed in three different colours (control sequences, environment names and comments). These are duplicated three times as 'alternate' glyphs (named `A.alt1`, `A.alt2`, `A.alt3`, etc.). Similarly, other glyph alternates are created for the number of different colours required. The default glyph for an alphabet remains uncoloured to follow the application or system colour theme. FontForge was used to create the glyphs; and duplicated glyphs use references of base glyphs. UFO3 font sources are generated by FontForge.

The `fontmake` open source tool supports generating colour (and variable) fonts in different formats: TTF and OTF. The `fonttools` [8] open source tool is used to create WOFF2 (web font format) files from the generated OTF.

`fontmake` supports generating both COLRv0 and COLRv1 formats, but their source specifications differ. The colour palettes and colour index association (colour layers) for each glyph are specified in a property list named `lib.plist`. The colour palette requires each colour to be specified in BGRA format. A sample of both formats can be found in Listings 1 and 2. The complete `lib.plist` is available in the source repository [7].

[3] Standards: `xkcd.com/927/`

[4] CPAL specification: `learn.microsoft.com/en-us/typography/opentype/spec/cpal`

[5] COLR specification: `learn.microsoft.com/en-us/typography/opentype/spec/colr`

[6] Unified Font Object: `unifiedfontobject.org/`

[7] M+ fonts: `mplusfonts.github.io/`

```
<!-- Colour layers -->
<key>
com.github.googlei18n.ufo2ft.colorLayers
</key>
<dict>
  <!-- 3 colours for each alternate alphabet-->
  <key>A.alt1</key>
  <dict>
    <key>Format</key>
    <integer>1</integer><!--PaintColorLayers-->
    <key>Layers</key>
    <array>
      <dict>
        <key>Format</key>
        <integer>10</integer> <!--PaintGlyph-->
        <key>Glyph</key><string>A.alt1</string>
        <key>Paint</key>
        <dict>
          <key>Format</key>
          <integer>2</integer><!--PaintSolid-->
    <key>PaletteIndex</key><integer>0</integer>
        </dict>
      </dict>
    </array>
  </dict>
  <key>A.alt2</key>
  <dict>
    ...
    <key>PaletteIndex</key><integer>1</integer>
  </dict>
  ...
  <!-- Single color for numerals -->
  <key>zero</key>
  <dict>
    ...
    <key>PaletteIndex</key><integer>3</integer>
  </dict>
  ...
</dict>
```

Listing 2: Defining COLRv1 layers for `fontmake`.

2.2.2 Contextual alternate lookups

In our font's highlighting of TeX syntax, control sequences, environment names, special characters, etc. are coloured as explained in previous section. Each colour (e.g. red brown for control sequences) is assigned to the colour layer of a glyph (e.g. `A.alt1`). The approach to colouring control sequences is explained in the following paragraphs. Recall that a TeX 'control sequence' is a sequence of letters preceded by the escape '\' character — e.g. `\LaTeX`, and non-alphabetic characters cannot be a part of it.

The 'contextual alternate' (`calt`) OpenType lookup can be used to code the syntax rules. The logic for colouring TeX control sequences is described in Listing 3, written in Adobe OpenType feature format. The apostrophe next to a glyph or glyph

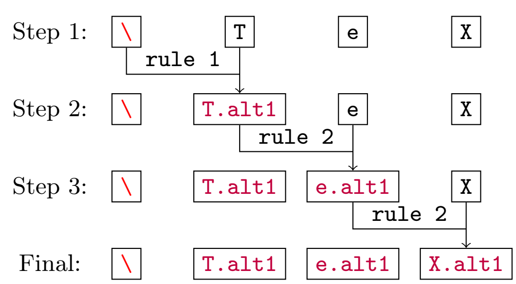
```
# Glyph ‘classes’, can have ranges of
# uniformly named glyphs
@TeXLetters = [ A-Z a-z ];
@TeXLettersAlt1=[A.alt1-Z.alt1 a.alt1-z.alt1];

feature calt {
  ...
  lookup TeXControlSequence {
    # Glyphs with apostrophe will be replaced
    sub backslash @TeXLetters’ by
                   @TeXLettersAlt1; # rule 1
    sub @TeXLettersAlt1 @TeXLetters’ by
                   @TeXLettersAlt1; # rule 2
  } TeXControlSequence;
} calt;
```

Listing 3: `calt` lookup for TeX control sequences.

class in rules 1 and 2 signifies the glyph(s) to be replaced. Other glyphs before ('backtrack') and after ('lookahead') the marked glyphs are the 'context'. A class of glyphs can be specified in the source and target of replacement, in which case the number of glyphs and their order in both source and target classes must match exactly.

The escape character '\' is coloured red brown by default and does not need any alternate glyph. When a letter immediately follows it, that letter is substituted with its desired alternate (here, the glyph with suffix `.alt1`). The alternate glyph gets the colour from the assigned colour layer (palette index).

The next challenge is to handle a sequence of letters immediately following the escape character. This is made possible by the state machine in rule 2 (OpenType is Turing complete, although rather a hard one to program): If a letter is preceded by an alternate form (here, `.alt1`), substitute the current glyph with its alternate form — this implements the state machine. This state machine stops at any non-alphabetic character (such as numbers or space or symbols). An execution of this process for the input '`\TeX`' can be visualized as:



This shows the lookup rule applied from Listing 3 at each step of execution. All `.alt1` suffixed alphabet glyphs are assigned the same color (red brown), resulting in the desired syntax highlighting.

Similar logic is employed for the `\begin` and `\end` of environments. The logic handles optional arguments (coloured in gray) to the environments, e.g. `\begin{thm}[Pythagoras' Theorem]`; and variants of `\cite`, e.g. `\citep*[e.g.][see]{cite_key1}`. However, the colouring will fail if there is a space, line break, or other whitespace between characters (see limitations in Section 2.3).

The complete OpenType feature file is available at the path `sources/TeXSynHL-Regular.ufo/features.fea` in the source repository [7].

2.2.3 Building the font

Once all the ingredients to build the font are ready in source forms — glyphs of all characters and their alternate forms in UFO format, colour palette and colour layers, and OpenType feature file — the binary font can be built using `fontmake`. The organization of the UFO sources (directories and files) is:

```
FontName.ufo/
  ├── features.fea ................ OpenType rules
  ├── fontinfo.plist .................... metadata
  ├── layercontents.plist
  ├── lib.plist ...................... CPAL, COLR
  ├── metainfo.plist
  └── glyphs/
        ├── A_.glif  A_alt1.glif
        ├── A_alt2.glif  A_alt3.glif
        ├── ...
        ├── z.alt2.glif  z.alt3.glif
        ├── backslash.glif
        └── ...
```

The general form of a build call looks like:

`fontmake -u` ⟨*ufo source*⟩ `-o` ⟨*out format*⟩

This will compile the UFO sources into a specified output format: either TTF or OTF. Web fonts can then be generated using `fonttools` by invoking:

`fonttools ttLib.woff2 compress` ⟨*OTF input*⟩

A `Makefile` is also in the source repository to automate these processes. In addition to the COLRv1 format, a variant of `lib.plist` for COLRv0 format is also kept. A make target switches to this version to generate a COLRv0 font from the same sources.

2.3 Limitations and possibilities

The implementation has certain limitations and possibilities to be aware of.

- By virtue of OpenType shaping rules, colours spanning multiple lines are not possible — for instance, a line break or whitespace before the left curly brace in `\begin{thm}` will cause `thm` not to be coloured. The whitespace could be fixed by duplicating rules 1 and 2 in Listing 3, adding a number of `space` glyphs to the context, but it is not a particularly elegant solution.
- Instead of `calt` substitutions, the HarfBuzz WebAssembly (WASM) shaper[8] could bring more flexibility in programmability. It might be attempted in the future.
- The Safari browser (version 26 as of this writing), does not support COLRv1 fonts yet, on either macOS or iOS.
- XeLaTeX does not support colour fonts. LuaHBTeX (version 1.18.0 as of this writing) supports COLRv0 fonts using the HarfBuzz renderer, but COLRv1 is not supported yet.
- CSS provides the properties `font-palette` and `@font-palettevalues` to override the default colour scheme baked into the font. The effect of a sample colour overriding is shown in Fig. 2 using the CSS shown in Listing 4.

To adapt this approach for a different language or system with a different font and/or different colour

```
@font-palette-values --Bright {
  font-family: "TeXSynHL";
  override-colors:
    0 red,    /* control sequences */
    1 yellow, /* optional params, comments */
    2 pink,   /* environments */
    3 lime,   /* numbers */
    4 green,  /* math, operators, ...*/
    5 orange; /* delimiters */
}
.code {
  font-palette: --Bright;
  font-family: "TeXSynHL";
  color: white;
  background-color: #2c2c2c;
}
```

Listing 4: CSS to override color palette in font.

```
\documentclass[12pt]{article}
\begin{document}
\section{Introduction}
An OpenType COLRv1 font with \TeX syntax highlighting.
With math: $a+b^c_{d+c^2}-\eta$
\par
And environments
\begin{thm}[Pythagoras' Theorem]
Handle optional parameters \citet*[e.g.][here]{key_one}.
\end{thm}
Some \textit{text here}.
Some %comment $\abcd+efg$ on this line
\end{document}
```

Figure 2: Overriding the colour scheme using CSS in Listing 4.

[8] HarfBuzz WASM examples: `github.com/harfbuzz/harfbuzz-wasm-examples`

```
\documentclass[12pt]{article}
\begin{document}
$a+b^c_d+c^2-\eta$
\begin{thm}[Pythagoras' Theorem]
Handle optional param \citet*[e.g.][here]{key_one}.
\end{thm}
Some \textit{text here}.
Some %comment $\abcd+efg$ on this line
\end{document}
```

LuaTeX bug: should be gray

Figure 3: Verbatim code typeset by LuaHBTEX using our COLRv0 font. On the `\begin{thm}` line, we see that the non-alphabetic `'` character breaks the colouring, possibly a bug. On the `Some %comment` line, the rest of the comment is coloured, possibly due to the same limitation (cf. Fig. 1).

schemes, it is possible by following the steps outlined in Section 2 or the prior art [4]. In particular, one would need to:

- Duplicate the alternate glyphs n times for n different colours.
- Generate the UFO format of the font source (glyphs and metadata).
- Specify the colour palette with n colours in `lib.plist` file.
- Assign a colour index to each glyph in `lib.plist`.
- Use `fontmake` to generate the colour font.

3 Application support

COLRv1 fonts are supported by the major OS and browser platforms, except, as noted Safari on macOS/iOS. Some desktop publishers also have support. In GNU/Linux, FreeType (since version 2.11) and GUI toolkits Qt (since version 6.9) and GTK4 support colour fonts. In general, though, browsers are better equipped to support all the features of colour fonts.

Among TEX engines, LuaHBTEX (version 1.18.0 as of this writing) supports COLRv0 format fonts (using `Renderer=HarfBuzz` option to `fontspec`), but COLRv1 fonts are not supported yet (see Fig. 3).

4 Code availability

The font with built-in syntax highlighting for TEX documents is unimaginatively named 'TeXSynHL'. Only a 'Regular' variant of the font is created, though it could be extended with a bold and/or italic variants if desired. The font is licensed under SIL Open Font License.[9] Glyphs reused from the OFL-licensed M PLUS fonts are the copyright of its authors.

All the sources, build tools, test files and documentation are made openly available in the source repository [7].

[9] OFL: `openfontlicense.org`

Acknowledgements

We'd like to thank Heikki Lotvonen for the prior art and the detailed write-up, Coji Morishita for the excellent M PLUS Code Latin font, and Rahul Krishnan for testing the font and feedback.

We're grateful for the discussions with Karl Berry and editorial corrections by him and Barbara Beeton.

References

[1] E.A. Blomfeldt. Emoji in Qt 6.9. 2025. `www.qt.io/blog/emoji-in-qt-6.9`

[2] Google Fonts. `fontmake`: Compile fonts from various sources to binary. 2015. `github.com/googlefonts/fontmake`

[3] J. Hoffmann, C. Heinz, B. Moses. `listings` — typeset source code listings using LATEX. 2024. `ctan.org/pkg/listings`

[4] H. Lotvonen. Font with built-in syntax highlighting. 2025. `blog.glyphdrawing.club/font-with-built-in-syntax-highlighting`

[5] Microsoft. OpenType specification version 1.9.1. 2024. `learn.microsoft.com/en-us/typography/opentype/spec`

[6] G. Poore, K. Rudolph. `minted` — highlighted source code for LATEX. 2025. `ctan.org/pkg/minted`

[7] K.V. Rajeesh, C.V. Radhakrishnan. Source of font with built-in TEX syntax highlighting. 2025. `gitlab.com/rit-fonts/texsynhl`

[8] J. van Rossum. `fonttools`: A Python library to manipulate font files. 1999. `github.com/fonttools/fonttools`

[9] G. Williams, et al. FontForge, free and open source font editor. 2000. `fontforge.org`

◇ K.V. Rajeesh
Rachana Institute of Typography
Trivandrum, Kerala
`rajeesh (at) rachana (dot) org (dot) in`
`https://rachana.org.in`

◇ C.V. Radhakrishnan
Sayahna Foundation
Trivandrum, Kerala
`cvr (at) river-valley (dot) org`
`http://river-valley.com`
ORCID 0000-0001-7511-2910