Foundry-in-a-box: A proposal for frictionless font publishing and distribution

Abstract

This article presents a proposal for a standard for font publishing and distribution. Starting from an analysis of the font design process, we then focus on the latter steps that are involved: preparing the font for release and distribution.

The lack of documentation for techniques or best practises regarding these final steps results in a multitude of personal workflows which we identify as a blocker to the release of more fonts and a larger ecosystem of type foundries, preventing the ubiquitous potential of libre and open font design.

A practical solution is then presented and showcased, based on a font packaging format and some self-developed tools that take advantage of it to simplify the font publishing workflow, ranging from simple metadata editing to the generation of foundry websites.

Keywords

Software, code, open fonts, font publishing, standards

Introduction

Purpose

This article describes a proposal for dealing with what we believe are the shortcomings of the current font publishing workflow, or rather the lack of a consistent one. We will describe the type design process focusing on topics we find as major blockers for a wider reach and spread of font publishing.

Terminology

Libre type is used to refer to fonts that are published under open and/or copyleft licenses. The term *libre* will be used across this paper to refer to fonts published under the Open Font License[1] (OFL), as well as other works licensed under *copyleft* or *copyfree* terms, which allow for redistribution, modification and sharing.

This choice of terminology is rooted on the ambiguity that permeates other common terms such as *free software* or *open* source. This wording issue hides a much bigger debate that would not fit on these pages; a good introduction to this distinction is laid out in Richard Stallman's essay *Why Open Source misses the point of Free Software.*[2] In order to avoid the confusion that the interchange of the terms *free* and *open* usually brings, we opted for the unambiguous *libre*.

The problem

The libre type design ecosystem has flourished in the last years. First with the early Open Font Library[3] hub for open fonts, started in 2006, followed by smaller independent libre foundries such as The League of Moveable Type,[4] OSP Foundry[5] or the VTF Type Foundry[6].

In 2010, Google Fonts[7] (at the time Google Webfonts) came into the spotlight, kickstarting the rise of libre typography by sponsoring type designers around the world to produce more libre fonts, in a collection that is growing bigger everyday.

However, there hasn't been much recent activity outside the existing hubs and foundries. It is our belief that this is caused by the difficulties and friction involved in publishing a font. The act of packaging and publishing is an integral part of the type design process; however, we find the publishing process is much less represented in type design literature and practice than the act of designing typefaces. Packaging and publishing are not glamorous tasks, and we believe that the lack of a standard workflow for post-processing and distributing a font can be a significant bottleneck in font production, leading to few new players in the field.

Each of the above mentioned libre foundries has developed their own, custom workflow for publishing and distributing typefaces: OSP Foundry is currently shifting from a WordPress website to a web front-end, generated with a set of custom scripts,

from their own Git repository; the League of Moveable Type started with a static HTML website and has recently developed a GitHub-based system to accept outside contributions and automatically update the website with improved versions of the fonts. The Open Font Library project has tried to develop the tools and solutions to help easily publish typefaces and become a universal hub for libre fonts, but hasn't yet succeeded on this challenge.[*] The development of these custom systems takes significant effort and time (often months), and not having a unified answer to this workflow management problem has resulted in a wasteful duplication of efforts.

As design practitioners who are familiar with the process of type design, we are aware of the difficulties that come into play when a finished font needs to be packaged for publishing and distribution.

The type design process, from start to finish, consists of a set of different tasks. These range from the more evident — like research, sketching, conceptual and aesthetic concerns, font design execution —, to the less obvious — such as the license choice, documentation, metadata information, among others. These less obvious tasks relate closely to what we could call the post-production phase: getting the font ready for publishing and distribution.

After finishing the design of a font, as well as taking care of spacing and hinting, there are many small tasks involved before a public release:

- choose the **output file formats**
- pick a **license**
- set and validate **font metadata**
- **document** the work (README and FONTLOG)
- ensure **compatibility** with various systems
- create a **downloadable package file**
- **upload** everything to a server

[*] The Open Font Library website developement has been on hold since 2012, when the framework it is based in, Aiki, announced its end. Since then there have been many threads on the project's mailing list regarding desired new features[8], by type designers, and discussion of which framework to adopt[9].

In addition, if a designer is also running their own type foundry, there are additional tasks:

- **create new web pages** for the new font
- update the **website text**
- add **specimens and/or preview images**
- add **classification or category**
- **upload** to other font distribution sites, if applicable
- maintain the **website back-end**
- periodically ensure every published version and metadata are **up-to-date**

While each individual step is not complex by itself, it is the collection of those necessary steps that creates **friction**. This term was borrowed from the *Frictionless Data*[10] document, published by the Open Knowledge Foundation, and describes the aggregation of tiny tasks which, when taken together, constitute a roadblock that is hard to overcome because of the accumulated complexity. The non-existence of any standard ways to carry these tasks out makes the problem worse, with each designer and/or foundry devising their own preferred formats, methods and workflows for font post-processing and publication.

State of the Art

OFont

Raphaël Bastide addressed this problem in the talk *Unified Typeface Design*[11], in April 2013, and presented a proposal for a standard for libre type font publication.

Later, in 2014, Bastide developed OFont,[12] a Content Management System (CMS) for archiving typeface collections. The use case was the website Use&Modify,[13] featuring a personal selection of typefaces, organised and categorised for self-use.

The main purpose of this CMS is to transfer the task of organising and categorising font collections from the type designers and/or foundries to the font users.

Font Bakery

Font Bakery[1] is tool aimed at preparing fonts for publishing, according to the norms of the Google Fonts API. The project, organised by Dave Crossland, started in 2013 and aims to implement best practises from software development into type design. It works with fonts published in a Git repository and automates a series of common steps in design publishing process.

Debian software packaging

Debian, one of the oldest and stablest Linux distributions, has provided us with clues for structuring this proposal. Software for the Debian operating system can be found in a central repository. This repository is curated in a decentralised fashion by volunteers, organised in a clear hierarchy, that are in charge of reviewing software and packaging it for Debian.

The Debian package model decouples the roles of the developer and the packager. The developer is responsible for coding the software, whereas the packager is in charge of reviewing and preparing the code to be distributed and used. Thus, developers don't need to further burden themselves with the intricacies of packaging software for specific operating systems, and packagers are an important addition to the development flow by ensuring and certifying the integrity of software. This clear division creates a structured and standardised workflow for the release of software, minimising the chance for errors and ensuring a smooth collaborative workflow.

There are significant benefits in harmonising the way we publish and release libre fonts. Having a shared set of best practices would result in a more transparent and horizontal workflow that can help reinforcing and expanding the libre type community, while also facilitating and encouraging local or remote collaboration.

Our proposal

This proposal aims at aiding designers in the process of publishing libre fonts and starting their own type foundries, either collective or individual, and either public or private. By streamlining the process of publishing and distribution, the libre type scene can grow much faster. It would also hopefully become richer, as there would be

a multitude of voices feeding the discourse and practice of type design.

Decentralisation is at the core of our *Foundry-in-a-box* proposal. Font collections and repositories play a fundamental role in the libre type scene. Nevertheless, having few other choices besides Google Fonts to find libre fonts, its potential for ubiquity is cut short.

While Bastide's OFont work deals specifically with the generation of the web interface to present a set of fonts, our proposal focuses on defining a standard, application-independent format for making fonts available on the web. This format could then power foundries' back-ends, as well as supplement aggregation applications like Ofont.

We thus propose to provide a common way to package and distribute typefaces, which involves:

- a flexible format for font packages
- tools that take advantage of this format for validating, editing and publishing fonts, including a foundry creator.

We first need to define what a font package actually is; our proposed definition would be the collection of elements that are essential for releasing and distributing a font or font family: the font files, metadata, and extra information. It could be extended to include specimens, extended documentation and other miscellaneous data.

Choice of technology

Before proceeding, we will quickly outline our choice of tools, formats and frameworks, as well as a quick explanation of each.

Python has surfaced as the language of choice for font hackers for writing their FontLab, RoboFont or Glyphs plug-ins. FontForge, the libre font editor, provides a Python interface out of the box, making it possible to inspect and alter font files using code.

Python's base libraries also allow us to quickly use it to work with Extensible Markup Language (XML) and JavaScript Object Notation (JSON) files, making it the perfect choice for this kind of endeavour.

We chose to have the font files under the **UFO format**, developed by Letterror, is the most accessible font standard currently available, mostly because of its XML-based structure.

Version control systems are present in any developer's Swiss knife, and they're becoming more popular in the type design field as well. The ubiquity of **Git** made it a straightforward choice, as well as the popularity of GitHub and Gitorious as distribution platforms.

Format description

The *Font Package* format we are proposing is heavily based on Open Knowledge's *Data Package* standard. While this standard was proposed for the specific use case of dataset distribution, we believe many of its virtues can be tapped in order to develop a sane solution for the challenge of packaging fonts.

Our format proposal sticks to the simplest structure possible. A font package would thus contain:

- the actual font files
- the metadata for the font file or family, with the full meta-information about the font family, its features, authors and other relevant information.

Proposed format

Under the proposed format, a font package would be structured in the following way:

```
roboto/
   fontinfo.json — Font info and user-specified metadata
   README.md — Human-readable description and details in MarkDown format
   FONTLOG — Detailed description of the font's version history
   roboto-regular.ufo/— Font files in UFO format
   roboto-bold.ufo/— Multiple weights can be included in the same package
```

The key file that makes this a font package is **fontinfo.json**, which contains all necessary data to describe the contents. Here is a minimal example:

```
{
    "name": "roboto",
    "title": "Roboto",
    "git_url": "http://gitorious.org/dummy/roboto.git"
```

```
  "resources": [{
   {
    "weight": "Regular",
    "path": "roboto-regular.ufo"
   },
   {
    "weight": "Bold",
    "path": "roboto-bold.ufo"
   }
  }]
 }  {
  "name": "roboto",
  "title": "Roboto",
  "git_url": "http://gitorious.org/dummy/roboto.git"
  "resources": [{
   {
    "weight": "Regular",
    "path": "roboto-regular.ufo"
   },
   {
    "weight": "Bold",
    "path": "roboto-bold.ufo"
   }
  }]
 }
```

Packaging workflow

Font packages enable a simple workflow for publishing fonts.

1. Design the typeface and save it as a UFO file.
2. Add the user-specified metadata in `fontinfo.json` (author info, date, license, etc).
3. [Optional] Add a `README` file with an introduction and description of the font file.
4. [Optional] Add or edit the `FONTLOG` with the recent additions.
5. Validate the package to ensure everything is ready to publish.
6. Publish the package to the remote Git repository.

Editing and updating existing packages is also straightforward.

1. Update the typeface's UFO file.
2. Update `fontinfo.json`, `README`, `FONTLOG` if necessary.
3. Validate the package to ensure everything is ready to publish.
4. Log the changes in a Git commit message.
5. Publish the package to the remote Git repository.

Tools

A proof-of-concept command line application, *fib*, is being developed in order to deal with all of these steps. This forms the best base to later develop a desktop or browser-based Graphical User Interface (GUI) to create and edit font packages.

This tool was designed to have a simple and human interface, using simple commands to achieve the workflow purposes outlined above.

Inside a Terminal window, one can create a font package with the command `fib init`. After copying the typeface and editing the `fontinfo.json` file, the command `fib sync` will make sure the UFO file reflects the metadata specified inside the font info file. This allows the user to quickly edit and complement the font's metadata without having to open their font editor.

On each significant change, `fib` reminds the user to commit the change, ensuring proper version control best practices. When the package is ready to publish, the command `fib validate` makes sure every file is in order, pointing out any inconsistencies in case they exist. Finally, `fib push` uses Git to publish the new version to a remote Git repository.

Another relevant feature of *fib* is the ability to generate fonts in several output formats. For instance, in order to generate the necessary formats for a package to be used as a webfont, one just needs to type `fib generate ttf otf woff eot svg`.

Foundry generator

We're also developing a special mode of the *fib* tool for a use case that will be familiar to type designers: the ability to create and maintain a simple type foundry website, which we use for our own foundry project.

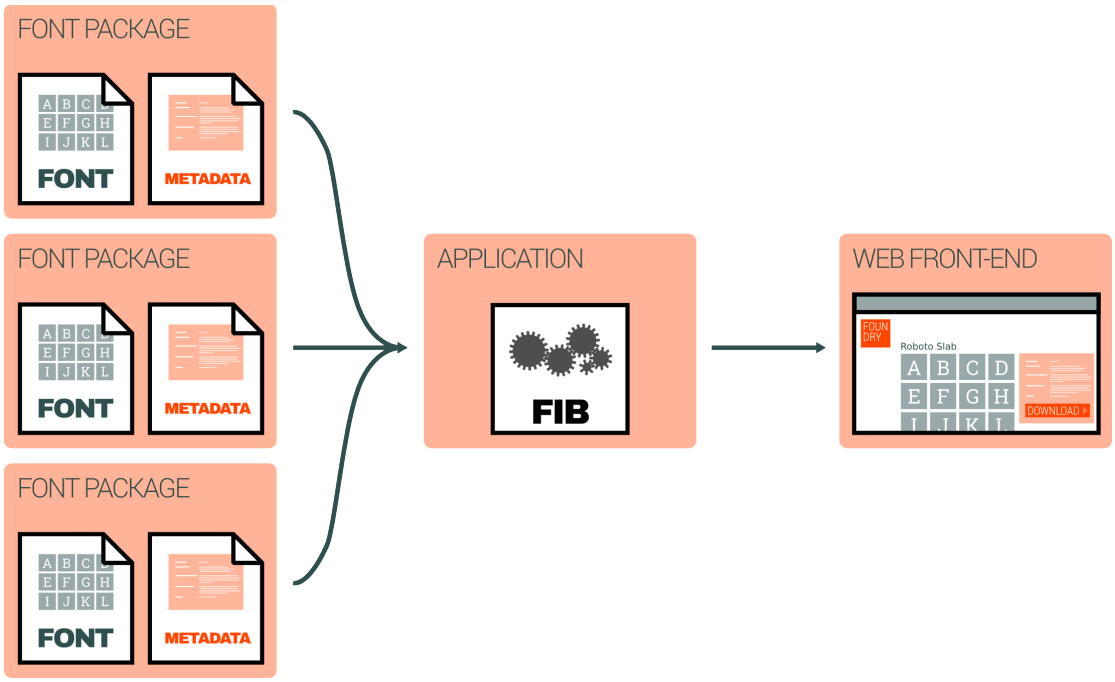

Figure 1: A fib-based web foundry workflow.

The generator is based on a list of fonts to include, as well as miscellaneous content (About and contact pages, for example) specified in MarkDown format.

```
foundry/
  - fonts.json
  content/
    - index.md
    - about.md
  templates/
    - base.html
    - index.html
    - page.html
    - font.html
  static/
    css/
    js/
    img/
```

This skeleton structure can be easily created with the command `fib foundry init`. The `fonts.json` file should contain a list of the font packages to be included in the generated site, and can be specified as locally stored font packages or remote Git repositories where they can be cloned from. Other *fib* commands are available in *foundry mode*, including `fib foundry validate`.

The generated site is plain HyperText Markup Language (HTML). While it was, for a moment, tempting to consider building a CMS tuned for type foundries, we have decided to not adopt any kind of dynamic features in this project, sticking to static HTML and, eventually, client-side JavaScript.

The site can now be uploaded with a simple File Transfer Protocol (FTP) client; however, if the destination's server properties are specified in `fonts.json`, the command `fib foundry upload` will take care of everything.

This way, we tried to minimise complexity as far as we could, and make sure any new user can get started with simple terminal commands and an FTP client in order to create their foundry website. This is the reason behind the static approach — not being the most glamorous or sophisticated option, it presents significant advantages:

- It is installable in any web server
- It requires, at most, superficial knowledge of FTP clients to upload the site.
- It does not require framework updates.
- It is the lightest possible solution when it comes to server resource usage.
- It is based on well structured and long standing standards.

This workflow proposal contrasts with the Google Fonts model in that it empowers designers to publish their own foundries without all the baggage that comes with the process; while Google Fonts centralises web font distribution, the *fib* foundry model aims for a distributed ecosystem of foundry sites.

Note about scope

The system we are proposing is geared towards the packaging and distribution of libre and open fonts. The libre font design community is where we come from, which is one of the reasons for us prioritising libre fonts over proprietary ones.

The font package format could be useful to proprietary foundries as an in-house interchange format, used as a common format to store and keep track of work. The short-term usefulness of this is not very significant, as most professional foundries already have their own workflows set up with their own choice formats.

On another hand, adapting the foundry-generation model to a proprietary workflow would not be complex: the only tweak necessary would be not to use any public facing resources. This means that a proprietary foundry could use *fib* as long as they set up a Git server for storing data packages. From there, *fib* can be used to generate the foundry site HTML.

Finally, we emphasise that proprietary workflows are not a priority to the goals of this project; while professional font publishers already have their own internal workflows, the libre community is still in need of a common solution to the packaging problem that we have described.

Conclusion and next steps

This proposal is a tentative first step to think of an important aspect of the type design workflow and fix some of its shortcomings — namely the less mentioned side of publishing a typeface online. The choice of technical solution draws from many reflections and debates we have had in the last two years, and is formulated as a way to eliminate the existing friction (and, therefore, frustration) in the process of packaging and distributing digital typefaces.

The font package format and the *fib* tool are an initial, effective way to provide a sane workflow for type designers and font packagers. We believe that these tools can be the foundation of more interesting developments, such as the improvement of this proposal into an actual standard.

References

[1] SIL International (2005), *Open Font License.* Published at http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL

[2] Stallman, Richard (2012), *Why Open Source misses the point of Free Software*, GNU.org: http://www.gnu.org/philosophy/open-source-misses-the-point.html

[3] Open Font Library (2014) [Webpage]. http://openfontlibrary.org

[4] League of Moveable Type (2014) [Webpage]. https://theleagueofmoveabletype.com

[5] OSP Foundry (2014) [Webpage]. http://ospublish.constantvzw.org/foundry

[6] VTF Type Foundry (2014) [Webpage]. http://velvetyne.fr

[7] Google Fonts (2014) [Webpage]. http://google.com/fonts

[8] From Vernon Adams (2012, December 25). [OpenFontLibrary] *The future of OFLB development*. Message posted to http://lists.freedesktop.org/archives/openfontlibrary/2012-December/003559.html

[9] From Daniel Johnson (2012, December 3). [OpenFontLibrary] *The future of OFLB development*. Message posted to http://lists.freedesktop.org/archives/openfontlibrary/2012-December/003544.html

[10] Pollock, Rufus (April 24, 2013). *Frictionless Data: making it radically easier to get stuff done with data*, Open Knowledge Foundation Blog. http://blog.okfn.org/2013/04/24/frictionless-data-making-it-radically-easier-to-get-stuff-done-with-data

[11] Bastide, Raphäel (2013, April). *Unified Typeface Design*. Talk at the Libre Graphics Meeting 2013, Madrid. See also the *Unified Font Repository*: https://github.com/raphaelbastide/Unified-Font-Repository

[12] Bastide, Raphäel (2014): Ofont [Software]. Available from https://github.com/raphaelbastide/ofont

[13] Use&Modify (2014) [Webpage]. http://usemodify.com

[14] Crossland, D., Kashkin, M., & Volkov, V. (2014) Google Inc.: Font Bakery [Software]. Available from https://github.com/googlefonts/fontbakery