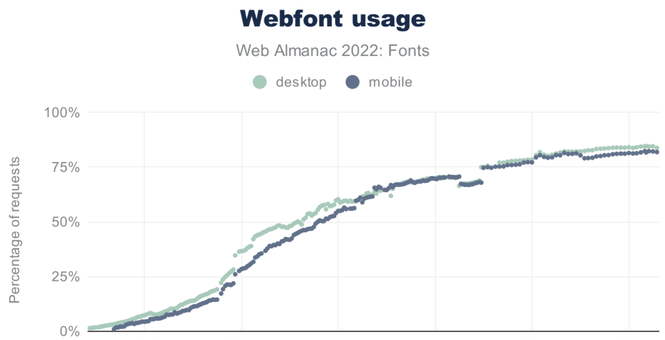
Prettybird: A DSL for Programmatic Font Compilation

Charles Averill
https://www.charles.systems/ charles@utdallas.edu
Co-Author: Takemaru Kadoi Advisor: Kevin Hamlen
The University of Texas at Dallas

Motivation

- Improvements in display technology and the adoption of the internet have led to greater public interest in **digital typography**. [7]
- Font design software and custom fonts have become more accessible and common, attracting beginners.
- Quality font design requires a **large time commitment**.
- Community lacks **beginner-friendly font design tools**, most are either **archaic or expensive**.



Designing tools that utilize the strengths of both graphical font design and font description languages provides existing font designers with more agency in designing unique fonts, and new font designers with a modern approach to the practice.

Background

METAFONT is an existing font description language developed by Knuth [4] [5] as a companion to TEX [3].

- METAFONTis comprehensive and versatile, but a very old piece of software initially burdened by hardware limitations [1].
- Can't generate **modern outline fonts** without the help of external post-processing [6].
- **Complex syntax** due to technological limitations of implementation languages.
- Relies on mathematical symbols and short keywords, making it seem archaic compared to modern description languages such as SVG and LATEX.
- Despite drawbacks, METAFONTis versatile and provides a high level of control with its **reusable pens** and **variable-length Bézier curves**.

Graphical font design suffers a separate set of problems, the primary issue being price. Of the five most-searched font design programs since 2011, **4 of them require a paid license**, and 3 of those licenses cost **$250 USD or more**. The one free program is also open-sourced, but development has slowed since 2014.

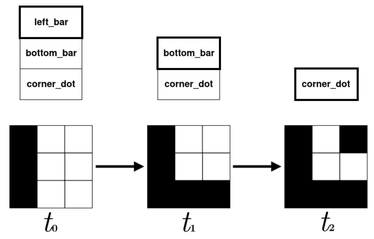
Compiler and Language Design

We present a compiler that generates **BDF, SVG, and TTF font files** from Prettybird source code. The compiler back-end additionally outputs **curve data viewable within the FontForge GUI**, for quick and seamless visual feedback to users. The figures below show an example outline font built with Prettybird, as well as a function used in the B, D, P, and R glyphs.


```
define bubble(top, bottom, width, direction) {
    // inner bubble
    draw bezier(top, top + (direction * width, width),
                top + (direction * width, 0))
    draw bezier(top, top + (direction * width * 1.25, width),
                top + (direction * width * 1.25, 0))
    // outer bubble
    draw bezier(top + (direction * width, width), bottom,
                bottom + (direction * width, 0))
    draw bezier(top + (direction * width * 1.25, width),
                bottom,
                bottom + (direction * width * 1.25, 0))
}
```

The language design was primarily motivated by a desire for readability, for both programmers and artists. This resulted in a syntax consisting of unabbreviated keywords and verbose, explicit block declarations. Prettybird supports basic mathematical operations, but complex behavior is defined only through recursion.

Each operation in the language modifies a **glyph space**, a 2D plane on which shapes are drawn to construct glyphs. Calling functions spawns a new, empty child glyph space on which that function's operations within that function operate. When the function terminates, its glyph space is combined with the space below it in the **glyph space stack** via a binary union.



Type System

We present a **novel type system** consisting of two types (numbers and pairs) that is functionally equivalent to METAFONT's 8-type system. The compiler back-end also implements a **type-checker** that implicitly enforces type-correctness of atom arguments.

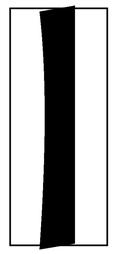
Anonymous User Survey

We present an **anonymous user survey** displaying user preference of Prettybird over METAFONT. **63 respondents** were provided code samples in both Prettybird and METAFONT [2] and were asked to choose which code sample was more readable, which they would prefer to modify, and which glyph looked better. They were also asked to self-identify as one or multiple of programmers, font designers, and familiar with Bézier curves. Respondents overwhelmingly preferred the simplicity of Prettybird's code and the higher-resolution glyphs it could produce, by margins of **upwards of 80%**.

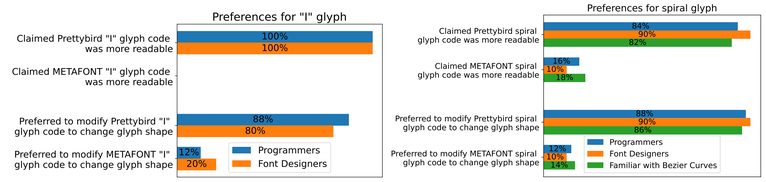
```
char I {
    base {
        blank(48, 72)
    }

    steps {
        draw vector((24, 6), (24, 42))
        draw bezier((18, 7),  (18, 43),
                (20, 24))

        draw vector((18, 7), (24, 6))
        draw vector((18, 43), (24, 42))
    }
}
```



One of the code samples and generated glyphs shown to respondents



Preferences of Prettybird over METAFONTin readability and modifiability among programmers, font designers, and those familiar with Bézier curves.

Reflection and Future Work

Prettybird provides new opportunities for font designers to approach the craft from a different angle by revitalizing an abandoned approach. We are currently improving Prettybird by better **integrating it with visual font design software** in order to bring a fresh approach to the field. Additionally, we are extending the language within the bounds of simplicity that we've set in order to provide a more powerful experience without sacrificing the ease of use the language is built on.

References

[1] Nelson H. F. Beebe. The design of tex and metafont: A retrospective. 2005.
[2] Jeremy Gibbons. Dotted and dashed lines in metafont. 02 1970.
[3] Donald Knuth. *TEX and METAFONT*. American Mathematical Society, 1979.
[4] Donald Ervin Knuth. *Computers and typesetting*, volume D - METAFONT: The Program. Addison Wesley, 1990.
[5] Donald Ervin Knuth. *Computers and typesetting*, volume C - The METAFONTbook. Addison Wesley, 1990.
[6] Han-Wen Nienhuys. mftrace, Dec 2011. URL `http://lilypond.org/mftrace/`.
[7] Bram Stein. Webfont usage. *Fonts | 2022 | The Web Almanac by HTTP Archive*, Sep 2022. URL `https://almanac.httparchive.org/en/2022/fonts#fig-1`.