

Digital Typography in a Multiscriptal World: Towards Fully Programmable Fonts

Kevin Graaf

5 February 2026

Keywords

digital typography, font design, stroke fonts, programmable typography, internationalization

1 Introduction

Dubsar, after the Sumerian word for *scribe*, is a new programmable typography engine and digital font format [2, 3]. It follows in the venerable tradition of parametric typography as pioneered by Knuth’s METAFONT [9], but improves upon this design in several key respects. Most significantly, *Dubsar* collapses all features of digital typography (e.g. kerning, shaping, ligature substitution) down to a single mechanism: programmatic glyph drawing. The generality of the *Dubsar* model enables it to effectively support multiscriptal digital typography, especially of non-Western, non-alphabetic, minoritized, and ancient scripts.

2 Full Programmability

Despite the increasingly global state of computing, considerable challenges remain in handling multiscriptal, multilingual, grapholinguistically *entangled* text [14]. OpenType, the current state of the art in digital typography, attempts to predefine all categories of typographic behavior through *shaping models* [10]. Even to the extent this is possible, this decision necessarily centralizes and hierarchicalizes how writing systems are computerized. These difficulties are exacerbated by the linear nature of Unicode, which



Figure 1: Mixed Chinese-Manchu Writing from the Forbidden City (source: Wikipedia Commons)

flattens the inherent structure present in many kinds of text [4]. Different scripts in a multiscriptal setting can typographically interact in surprising and hard to predict ways. When one considers the combinatorial explosion of script pairings, it becomes obvious that it is simply infeasible if not impossible to enumerate every kind of interaction.

I argue for a new paradigm of digital typography—*programmable typography*—which decentralizes and democratizes computer text by providing a single mechanism of sufficient generality to handle essentially all human writing systems both individually and in entangled, multiscriptal contexts.

3 Dubsar Programming Model

While METAFONT fonts are *parametric*, they are not *programmable* in the fullest sense as METAFONT generates bitmap renderings of glyphs *statically* and offline. Dubsar instead conceptualizes fonts as *functions* from an abstract, yet structured representation of text (see figure 2) to concrete, graphical forms. Indeed, Dubsar fonts are merely programs which target the Dubsar Virtual Machine, a simple, stack-based bytecode closely related to the Forth programming language [12]. These font-programs fully represent the underlying “logic” of any writing system within their coverage in a self contained manner, without recourse to pre-defined shaping models.

By contrast to the monumental complexity of OpenType and other related technologies, the ability to correctly display and interactively edit text in any writing system for which a Dubsar font has been provided is fully encapsulated within the font’s code and is automatically conferred upon all software applications which implement the relatively simple Dubsar Virtual Machine.

4 Graphics Model

Digital fonts may be roughly divided into three graphical categories. Raster or bitmap fonts represent glyphs in terms of discrete pixels. Outline fonts specify the outline of glyph shapes as splines such as Bézier curves. Stroke fonts procedurally describe the construction of a glyph through the placement of strokes, in a manner roughly analogous to how characters are written by hand.

Despite their relative obscurity, stroke-based formats have considerable advantages over outline and raster fonts, especially for CJK characters [1]. In particular, they retain the resolution independence of outline fonts, yet

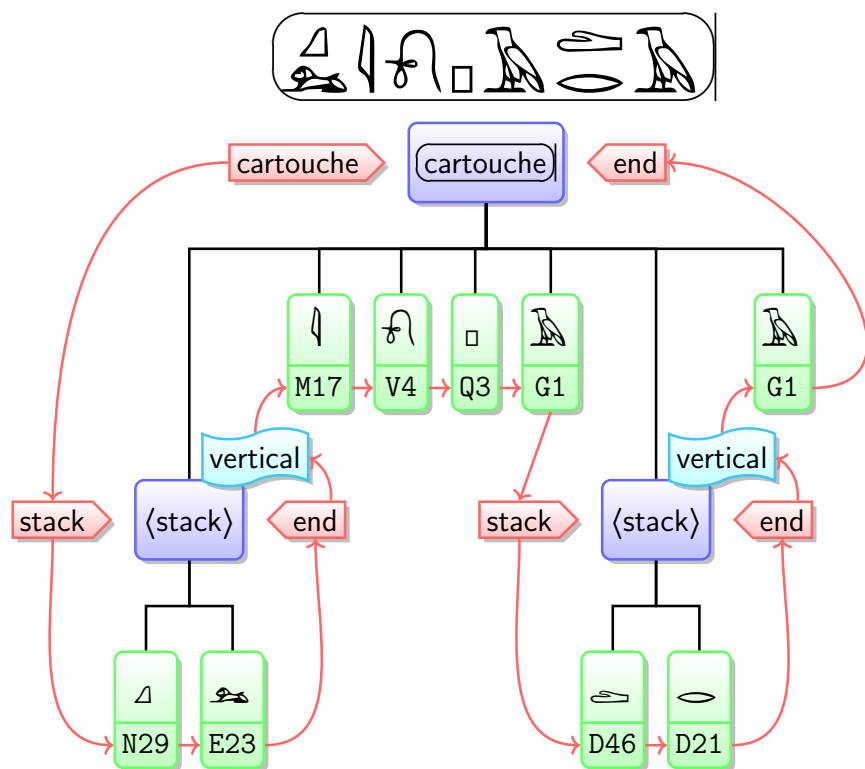


Figure 2: Example of a structured representation of Egyptian Text

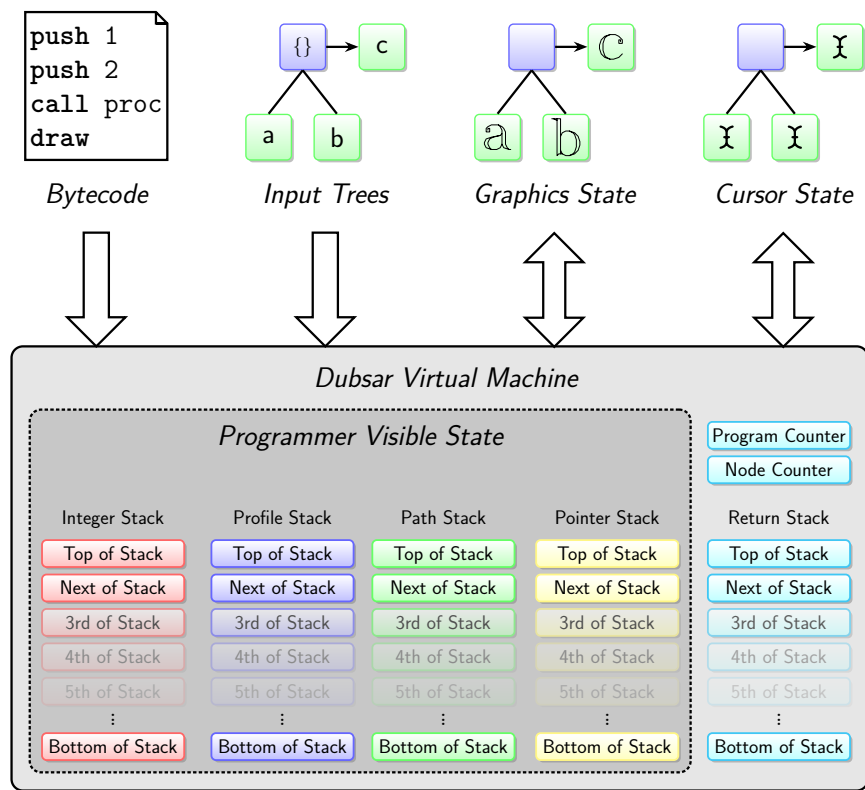


Figure 3: Schematic of the Dubsar Virtual Machine

provide a more compact and "semantic" representation of glyphs which minimizes the need for manual hinting [8].

Stroke fonts may be further subdivided into pen-based and profile-based graphical models. In the former category, typified by METAFONT, the outline of a virtual pen nib is specified along with a stroking path. Rendering graphics in the METAFONT pen model requires significant overdraw, as the pen shape must be "stamped" at every (discretized) point along the stroking path. This is sufficient for offline bitmap font generation, but is highly inefficient for direct, online rendering of text.

Profile-based stroke fonts represent a glyph stroke as a *centerline* and one or more one-dimensional profile splines which specify the perpendicular width of the stroke at any given point along it. The Dubsar graphics model builds on previous profile-based graphics models, in particular Klassen's variable width splines [8], Hutchinson's *Stylized Stroke Fonts* (SSF) [5], and Jagenstedt's *Vector Graphics Stylized Stroke Fonts* (VGSSF) [6], by introducing a rotational degree of freedom: the edge of a stroke is no longer reckoned as a perpendicular distance from the centerline, but rather as the distance at an angle determined by an additional *rotation* spline. This feature both simplifies Dubsar's graphics model over its predecessors and sidesteps much of the difficulty associated with stroking Bezier curves at cusps [13, 7] by simply directly specifying the intended behavior.

5 Paper Overview

In this paper, I argue that *programmable* typography, as I have defined this term, is uniquely suited to meeting the challenges of multilingual and multiscryptal typography, yet can also dramatically decrease the net complexity of typography software.

This paper will give a brief history of programmable typography and stroke-based typographic graphics models, contextualizing Dubsar's design. The paper will then, for the first time, provide an extensive technical description of Dubsar and its implementation and will highlight two profound advances over previous iterations of the system, 1) a pattern matching based execution model inspired by the Standard ML programming language [11] and 2) an entirely new graphical model which constitutes a significant improvement on the state of the art.

In addition, my paper will also illustrate the difficulties of adapting digital typography to heretofore neglected scripts through the case study of Maya, a script which shatters Western expectations of what writing can

be. I detail the development of an experimental Maya font created with the Dubsar system and highlight the challenges presented by the nascent field of Maya typography.

References

- [1] Joel Breckinridge Bassett. *Stroke Fonts Forever*. July 18, 2021. URL: <https://atadistance.net/2021/07/18/stroke-fonts-forever/>.
- [2] Kevin Graaf. “Carving Text at Its Joints: A New Perspective on Writing and Computers”. In: *Proceedings of the 2025 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! ’25. Singapore, Singapore: Association for Computing Machinery, 2025, pp. 194–203. ISBN: 9798400721519. DOI: 10.1145/3759429.3762630. URL: <https://doi.org/10.1145/3759429.3762630>.
- [3] Kevin Graaf. *Dubsar: A New Approach to Rendering Non-Western and TEI-Encoded Text*. 2025. URL: <https://ftp.kbg.sh/papers/tei2025-20250306.pdf>.
- [4] Yannis Haralambous and Martin Dürst. “Unicode from a linguistic point of view”. In: *Graphemics in the 21st Century*. Fluxus Editions. 2018, pp. 167–183.
- [5] Elena Jakubiak Hutchinson. “An improved representation for stroke-based typefaces and a method for their creation”. PhD thesis. Tufts University, 2009.
- [6] Philip Jägenstedt. *Vector Graphics Stylized Stroke Fonts*. Tech. rep. 2008. URL: <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A18417&dswid=5338>.
- [7] Mark J. Kilgard. “Polar stroking: new theory and methods for stroking paths”. In: *ACM Trans. Graph.* 39.4 (Aug. 2020). ISSN: 0730-0301. DOI: 10.1145/3386569.3392458. URL: <https://doi.org/10.1145/3386569.3392458>.
- [8] R. Victor Klassen. “Variable Width Splines: a Possible Font Representation?” In: *Electronic Publishing* 6 (1993), pp. 183–194.
- [9] Donald E. Knuth. *The METAFONT book*. United States: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0201134446.

- [10] Microsoft Typography. *OpenType Specification*. Tech. rep. Version 19.1. May 2024. URL: <https://learn.microsoft.com/en-us/typography/opentype/spec/>.
- [11] Robin Milner et al. *The Definition of Standard ML*. The MIT Press, May 1997. ISBN: 9780262287005. DOI: 10.7551/mitpress/2319.001.0001.
- [12] Charles H Moore. “FORTH. A new way to program a mini computer”. In: *Astronomy and Astrophysics Supplement* 15 (1974), p. 497.
- [13] Diego Nehab. “Converting stroked primitives to filled primitives”. In: *ACM Trans. Graph.* 39.4 (Aug. 2020). ISSN: 0730-0301. DOI: 10.1145/3386569.3392392. URL: <https://doi.org/10.1145/3386569.3392392>.
- [14] Iair G. Or. “Typographical advocacy in the age of digital encoding”. In: *Language Policy* 21.4 (Dec. 2022), pp. 527–543. ISSN: 1573-1863. DOI: 10.1007/s10993-022-09619-2. URL: <https://doi.org/10.1007/s10993-022-09619-2>.