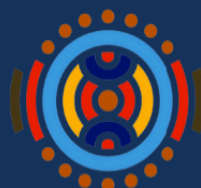


Indigenous Languages:

**Zero to Digital**

# Font Creation Guidelines

TRANSLATION  
COMMONS



2022-2032 | INTERNATIONAL DECADE OF  
**Indigenous Languages**

Translation Commons © 2025

This work is licensed under a Creative Commons Attribution 4.0 International License

# Table of Contents

<b>Introduction</b>	<b>4</b>
How to Use These Guidelines	4
Key to the Figures	5
The Components of Writing	6
Base Characters	6
Diacritic Marks	6
Numbers	7
Special Symbols	7
Punctuation	8
Alternate Character Forms	8
Positional Forms	9
Ligatures	9
Important Note	10
Getting Started	11
Assessing Font Requirements	11
Initial Steps	11
Vertical Metrics	11
Designing Glyphs	12
Sketching	12
Establishing the Overall Traits of Your Design	12
Setting Glyph Space	14
Connected Scripts	15
Complex Shaping	16
Font Editing Software	17
OpenType	17
Substitution	17
Position	18
Complex Shaping	18
Auto-generated Code	18
Anchors and Combining Marks	19
Testing Your Font	21
Collisions	21
Legibility	22
Shaping	23
Advanced Topics	24

Design Concepts	24
<b>In Summary</b>	<b>29</b>
Appendix: Digital Font Design Editors	30
<b>Appendix: Glossary</b>	<b>31</b>
<b>Appendix: Resources</b>	<b>34</b>
Tools	34
Further Resources	34
<b>Acknowledgements</b>	<b>36</b>

# Introduction

This document is one in a series of guidelines entitled *Zero to Digital* addressing language digitization practices, authored by Translation Commons. The entire series is available at <https://translationcommons.org/resources/>.

## Audience

This guideline offers an overview of the font creation process and step-by-step instructions. It is created specifically for Indigenous language communities intending to digitize their language. This guideline is written for individuals without advanced skills in computer technology or training in linguistics. That said, having some font development experience, or teaming up with others who do, can accelerate creating a font that will be successful.

## Goals

This guideline is intended to enable readers to create a font for their language that will be **legible, functional, and acceptable to the community**.

In particular, the aims of this document are to:

- Provide an overview of the differences between writing systems;
- List the types of language data that must be collected to design a font;
- Describe a process for creating a font that meets the language requirements; and
- Enable readers to produce a basic, functional, and useful font.

High-quality fonts make use of very nuanced attention to positioning, shaping, and sizing of glyphs as well as the relationship of each character to its surrounding context. The resources required to describe how to achieve that level of quality is beyond the scope of this document. Interested readers can see more detailed material in the reference section.

## How to Use These Guidelines

Creating a font is a mix of linguistic, technology, graphical, and people skills. The first few sections provide background that you need to know about digitizing and visualizing text. This is intended to help you assess the requirements that your font must satisfy for your language. From there, this document will introduce font editing and demonstrate with examples. There is also a section on testing you can perform to assure the quality of your font.

Designing and creating a font is generally an iterative process. A first step is to create sketches and establish designs for your font. You can create a basic font, and as it is

used with more text, you and your user community will discover its limitations. You will then make further improvements, and the cycle repeats. This document will also touch on some advanced font topics with examples that can help with more nuanced issues.

## Key to the Figures

This document uses the following symbols in the figures/illustrations:

**Dotted Circle (⦿):** a placeholder for a base glyph. Shows the placement of a combining mark in relation to the base character. The base character is represented by the dotted circle.

**Plus symbol (+):** indicates an input sequence of glyphs and combining marks.

**Arrow (→):** indicates the transition from one form to another after applying a function, or a change in form when applying a new design idea.

# The Components of Writing

Writing uses many kinds of **glyphs**. Letters or ideograms, often called characters, are glyphs, as are punctuation symbols, digits and arithmetic symbols, diacritic and tone marks, etc. In other words, any shape in a font is a glyph, whether it's a letter, number or symbol. There are also special symbols, for example: currency, copyright, and others. This section describes these different glyphs and how they are used in fonts.

## Base Characters

The base characters are the fundamental units of any written language. These will make up the majority of the glyphs in your font.



Fig. 1 Base characters are the basic units of a writing system, whether it is a letter, syllable, or logograph (left, Adlam; middle, Ge'ez; right, Vai)

## Diacritic Marks

Many writing systems utilize marks that are placed either above, below, or alongside a character to modify its pronunciation or to indicate tone or distinguish it from similar characters. There are two kinds of diacritic marks:

**Combining marks** are diacritics or tonal marks positioned within the horizontal space occupied by a base character. When combining marks are attached to a base character, they generally do not add width to the character and are referred to as combining marks.

**Spacing diacritics** behave similarly to letters in that they appear alongside the affected letter when typed, occupying their own horizontal space.

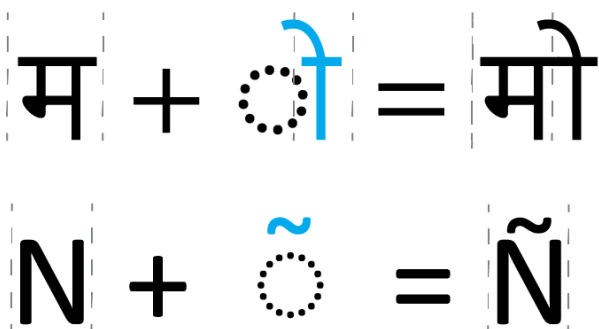


Fig. 2 Comparison of spacing diacritic (top) and combining mark (bottom). In the top example, the spacing diacritic (blue) abuts to the base letter. In the bottom example, the combining mark (blue) occupies the same space as the base letter.



Fig. 3 Combining Marks (blue) are elements that are applied to base characters. These are often placed directly above and below the base character, but also top-right in some writing systems.



Fig. 4 Combining Marks (blue) can also appear connected to the base character (Latin).



Fig. 5 Elements that are considered part of the base character (pink) are not treated as combining marks and are drawn as part of the character. Note that some cultures may differ on what is considered a base character and what is a combining mark.

## Numbers

Many writing systems have their own method of writing numbers. In some cases, numbers are borrowed from other writing systems. Whether native or borrowed from another writing system, it is important to include numbers when creating a font. Some writing systems may include ordinal numbers and special numeric symbols.

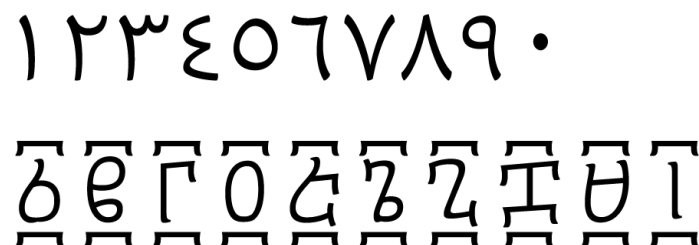


Fig. 6 Numbers from different writing systems. (top, Arabic; bottom, Ge'ez)

## Special Symbols

Special symbols are glyphs used for specific purposes but are usually written separately from the words in a language. These can include, but are not limited to: currency marks, mathematical operators, and other forms of notation.

@ # \$ + = 2 ɓ ɸ

Fig. 7 Left to right: “at” symbol, hashmark, dollar sign, plus sign, equal sign, oo Dennen (N’ko grammatical symbol), dorome-toosere (N’ko currency symbol), taman-toosere (N’ko currency symbol).

## Punctuation

Many writing systems have their own punctuation, use borrowed punctuation, or a mix of both. It is important to include all of the punctuation marks in your font. Be sure to consider anything that is required for various types of content, such as quotations, hyphens, etc.

، ؟ ﻻ ﻻ ( ) . , : ; “ ” « »

Fig. 8 Left to right: Arabic comma, Arabic question mark, Arabic ornate parentheses, common parentheses, period, comma, colon, semicolon, double quotes, guillemets.

**Base Characters, Diacritic Marks, Numbers, Special Symbols** and **Punctuation** are the elements required in a font in order to address the basic needs of a script or writing system in which the characters do not connect. Other considerations come into play for scripts or writing systems in which the characters are connected, or where alternate character forms are required.

## Alternate Character Forms

Alternate character forms have the same phonetic value or function as another character but have a different visual representation. Alternate characters are often required in a font to address regional or cultural preferences in the visual style of a writing system.

ɲ ʋ ɓ default  
ɲ ʋ ɓ alternate

Fig. 9 Alternate forms visually appear different from their default counterparts but have the same function.

Some writing systems require characters to change shape or combine with others depending on regional or orthographic preferences.



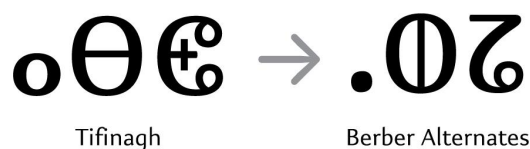


Fig. 10 The Tifinagh script is used in many linguistic and regional variations that require alternate forms. Left, three characters of the Tifinagh script; right, their Berber alternates.

## Positional Forms

Positional forms are variations of the base character that include connecting strokes and visual changes depending on where the letter occurs in a word.

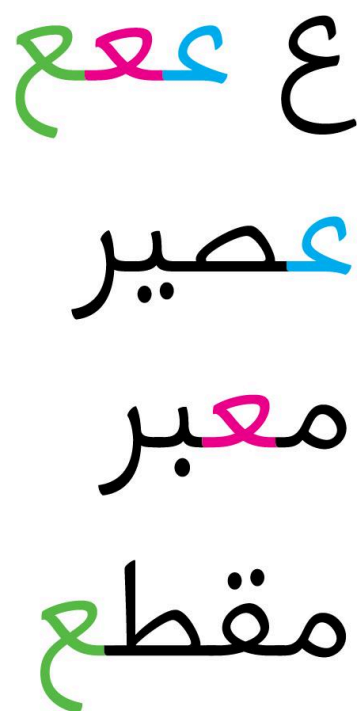


Fig. 11 Positional forms of the Arabic letter Ain as an example; initial (blue), medial (pink) and final (green) are variations of the base character (black) that include the connecting stroke and any visual changes that occur depending on where the letter occurs in a word.

## Ligatures

A ligature is a visual representation of multiple characters that are joined together in some fashion. In some scripts, such combinations are mandatory based on the orthographic rules of the writing system. In Arabic, the laam-alif ligature shown in the top row in figure 12 is a mandatory ligature, whereas the jeem-hah ligature in the bottom row is an optional ligature.

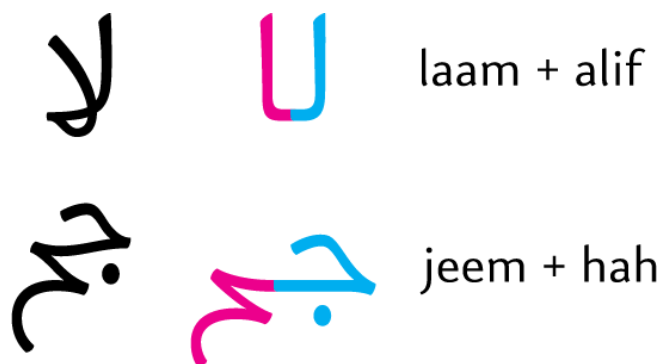


Fig. 12 Ligatures (black) are variations of character sequences that are joined together in a fashion that is different from the default behavior (blue and pink) of each individual character. (Arabic)

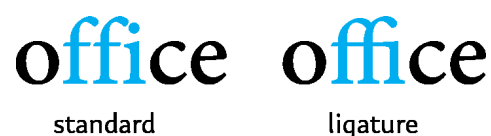


Fig. 13 Certain characters can join for a preferred appearance when set next to each other. (Latin)

## Important Note

Note that the implementation of glyphs in fonts can be influenced by the way glyphs are encoded and the choices made in laying out a keyboard for inputting those glyphs. The way a glyph renders on screen is dependent on the underlying properties of its encoding. These properties are assigned during the encoding process. For example, if a combining mark is set up with the wrong properties in your font, it will not attach properly to the base character. It is also important to consider how the script will be typed. Anything that a user may want to type needs to be assigned to a key on the keyboard. This can be challenging for a script with a lot of glyphs, given the limit on the number of keys available. This can be managed with [advanced features](#) that enable the font to automatically display glyphs that occur in specific contexts or combinations.

# Getting Started

## Assessing Font Requirements

There are significant differences in the writing systems used across languages. This results in different requirements for fonts used with each language. This section describes various font features applicable to many writing systems. Consider each feature and whether it applies to the font you are creating for your language.

### Initial Steps

The first step to creating a digital font for your writing system is to catalogue a complete list of glyphs that are needed. Glyphs can fall into different categories, such as letters, numbers, diacritics, etc. The following guidelines will help you determine how to classify your glyph types. When making the font, it is important that glyphs are associated with the proper glyph types in order for it to function as intended.

### Vertical Metrics

The next step in creating your font is to determine the space each individual glyph will occupy. These are called the **Vertical Metrics**. The vertical metrics create a kind of window, or **bounding box**, in which each glyph will appear.

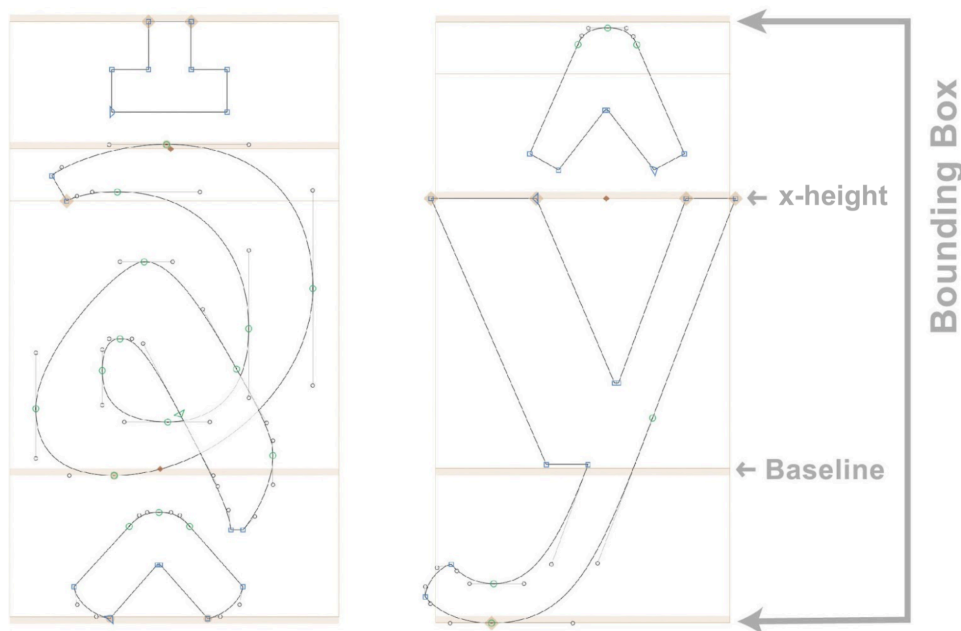


Fig. 14 The vertical metrics and bounding box as displayed in font editing software. The tan stripes represent alignment zones for two different scripts.

- 



- 



Also, if your writing system has recurring shapes, such as vertical strokes or round elements, a couple of characters containing these shapes should be included in the selection as well.



Fig. 17 Characters with recurring shapes. (Cherokee)

Working on this limited selection of glyphs will allow you to determine the basic characteristics of your font, such as weight (stroke thickness), width proportion (e.g. narrow, normal), contrast (change in stroke thickness), specific style (e.g. loop or non-loop in Thai, upright characters or leaning characters), and overall expression (e.g. angular, round, soft, hard).



Differences in stroke thickness



Differences in width proportion



Differences in contrast



Differences in style (non-looped & looped Thai)



Differences in overall expression

Fig. 18 Basic font characteristics.

When drawing the most complex forms, bear in mind how they may appear in small sizes. It's beneficial to keep complex forms as simple and open as possible. While the

examples in Fig. 18 show more than one font for each script to highlight some font style choices, you will start by adhering to one style in designing your font for your script.



Fig. 19 Left to right: a complex form made simpler, set in large and small sizes.

Once these basic characteristics have been established, you will have the criteria you need for creating and developing the remaining characters in your writing system.

## Setting Glyph Space

It is very important to carefully establish the spacing between the glyphs along with the character widths themselves, since the consistent rhythm of shape and space facilitates the reading process. In digital type, the spacing is represented by numerical values for the space to the left and to the right of a glyph. Establishing and adjusting these values is essential to the optimal spacing of the characters in a font.



Fig. 20 Adjusting values for optimal character spacing.

When creating a font for long texts, it is best to aim to make the space between forms *optically* equal in volume to the spaces within the forms. A rhythmic consistency in spaces will assure optimal readability.



Fig. 21 Strive for a volume of horizontal space between the characters that is close to the volume of space within the characters.

For scripts where the characters connect like Devanagari or Arabic, the letters should be spaced so that there is some overlap between the connecting strokes of each letter.

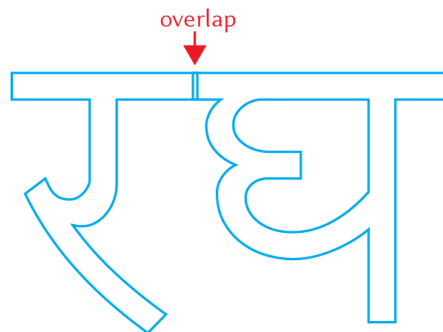


Fig. 22 Connected scripts are spaced such that there is a small overlap between the characters. (Devanagari)

## Connected Scripts

If your writing system is one in which the letters are connected (e.g. Arabic), letters may have up to four kinds of contextual alternates: Characters that do not connect to other characters (isolated), characters that connect on the left side only (initial), characters that connect on both sides (medial), and characters that connect on the right side only (final). Depending on your script, the shape of the letter may change depending on where the character occurs in the word.

In the following figures 23 and 24, note that the Arabic and Adlam languages are written from right to left. Therefore, the initial character is located in the right-most position.



Fig. 23 Connected scripts (left, Arabic; right, Adlam) have letters that connect to adjacent letters. This requires four different drawings. Isolated (black), initial (blue), medial (pink), and final forms (green).



Fig. 24 The letter Ain in the Arabic alphabet in its isolated form (black), initial form (blue), medial form (magenta) and final form (green).

## Complex Shaping

Some writing systems have very complicated systems that involve many complex form changes and rearranging of glyphs.

Input Sequence	Output	Shaping
र Ra	र	
् Virama	र	Virama attached below
द Da	द	Ra changed to Reph and attached above Reph is the above-based form of Ra Virama removed
् Virama	द	Virama attached below
म Ma	म	Da-Ma conjunct formed Virama removed Reph repositioned
ि I-Vowel	मि	I-Vowel reordered before syllable

Fig. 25 In complex scripts glyphs are re-ordered and change shape as characters are input. (Devanagari)



# Font Editing Software

Special font editing software will be used to produce a font. There are many options available with different capabilities, price points, and skill level requirements. The most common editors can be found in the [Appendix: Digital Type Design Tools](#).

## OpenType

OpenType is the industry standard programming language that enables a font to correctly form and position characters in various contexts.

Let's start by recognizing that you may want to define particular operations (known as rules) to be performed upon request by users. The glyphs to be affected, the set of rules to be applied, and the request that triggers the operation are defined in what OpenType calls a feature.

[Features](#) are used by the text rendering engine to execute the specified rules under the right conditions. OpenType performs two kinds of rules. Rules can substitute glyphs for the original glyphs, and they can reposition glyphs. You can define, for example, a feature for replacing glyphs with small caps, or a feature to replace a sequence of glyphs with a ligature, or a feature to replace a glyph with an alternate glyph.

The following sections introduce some of the OpenType rules.

An in-depth discussion of OpenType programming can be found [here](#).

## Substitution

Glyph substitution is an operation where one or more glyphs in the font are replaced by others. Below are examples of some OpenType rules for substitution.

```
sub <glyph name> by <glyph name>;
```

Example of single glyph substitution where one glyph is replaced by another.

```
sub <glyph name> <glyph name> by <glyph name>;
```

Example of ligature substitution where a sequence of two or more glyphs are replaced by a single glyph.

```
sub <glyph name> <glyph name>' by <glyph name>;
```

Example of context-dependent substitution where the second glyph is replaced only if it follows the first glyph. Note that the tick mark indicates which glyph is being replaced.

## Position

Glyph positioning is an operation where the position of a glyph is changed from its default location. Typically, one does not need to write glyph positioning code as it is automatically written by the font editing software. Oftentimes, if there is a need to reposition a glyph, it is more intuitive to draw an alternate glyph having the adjustment and use a substitution feature instead.

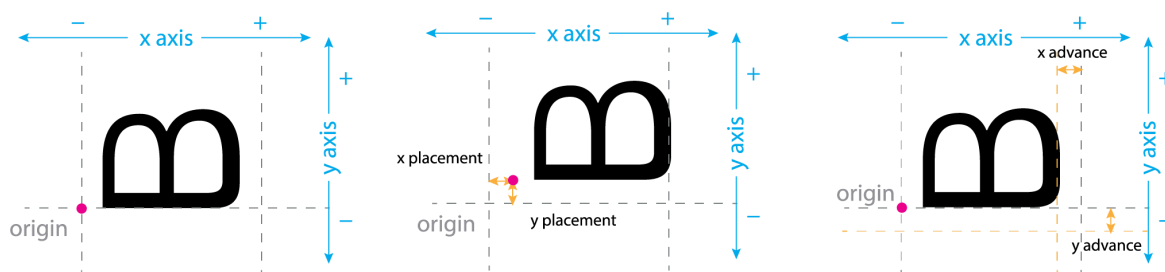


Fig. 26 Glyph positioning. (N'ko)

Below are examples of some OpenType rules for glyph positioning.

```
pos <glyph name> <xPlacement yPlacement xAdvance yAdvance>;
```

Example of moving a glyph within its bounding box. Placement values move the outline within the bounding box, and advance values resize the bounding box.

```
pos <glyph name> <glyph name> <value>;
```

Example of spacing adjustment between two glyphs.

## Complex Shaping

Indic shaping is quite complicated and therefore is outside of the scope of this document. It is not required by the majority of writing systems. The following links provide explanations on how these features work:

- [Developing OpenType Fonts for Devanagari Script](#)

This document targets developers implementing Indic shaping behavior compatible with Microsoft OpenType specification for Indic scripts.

- [Indic Script Shaping in OpenType](#)

This document outlines the general shaping procedure shared by all Indic scripts and defines the common pieces that script-specific implementations share.

## Auto-generated Code

Modern font editors can automatically generate OpenType code, provided you name your characters in a certain way. This is a convenient tool that reduces the learning

curve in implementing [OpenType features](#). The examples below illustrate how code can be automatically generated in most modern font editors. It is best to consult the user manual for the chosen font design editor for more precise instructions.

- For diacritic placement, simply defining anchors properly will automatically create the code for the [features](#).
- Similarly, setting up kerning in the font editor will automatically create the [kern](#) feature.
- To create a ligature feature, the ligature character is typically named using an underscore. For example, to create a ligature feature for *f* and *i*, name it: ***f\_i***.
- Alternate forms are typically named with the ***.alt*** suffix. For example, an alternate form of the Tifinagh letter *yagh-tifi* would be named ***yagh-tifi.alt***.
- Positional forms are named with a suffix corresponding to the position. For example, positional forms for *kpo-adlam* are named ***kpo-adlam.init***, ***kpo-adlam.medi***, and ***kpo-adlam.fina*** for the initial, medial, and final forms, respectively, in the Adlam script.

## Anchors and Combining Marks

In digital font design tools, anchors are used to control the positioning of diacritics and combining marks. This is regardless of whether they are joined to a letter or positioned near it. To position a combining mark, two anchors are needed: a receiving anchor on or near the base letter and an attaching anchor on the combining mark.

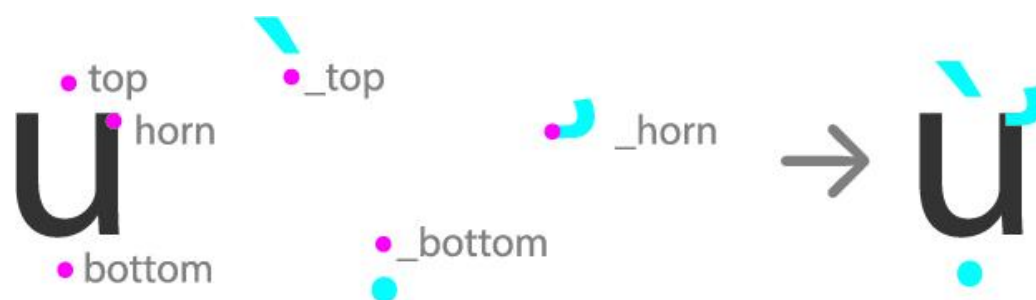


Fig. 27 Setting up receiving anchors (pink) for the attaching anchors on the combining marks (blue).

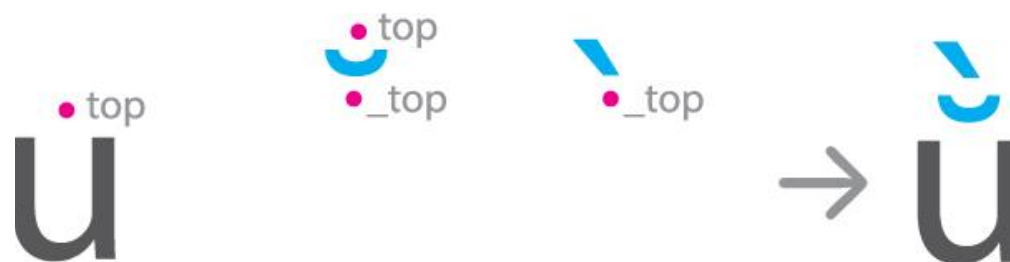


Fig. 28 Setting up receiving anchors (pink) for stacked combining marks (blue) placement.

It is technically possible to draw base glyphs together with combining marks as an individual character in your font rather than using mark attachment. However, this approach is generally not recommended, as each one of these combinations will then need to be assigned to a separate key on the keyboard in order to be able to type them.

A comprehensive discussion can be found in the [Glyphs Handbook](#).

# Testing Your Font

As you make your font, you will want to test it periodically to make sure it is working as expected. Some common points for testing are: control characters; basic character set; figures, punctuation and ligatures; and kerning. Consult your [font design editor's](#) handbook for instructions on installing your font.

Testing fonts requires examining their legibility in real-world usage scenarios such as signage, UI elements, and print materials. Font Bakery is a valuable free web application that evaluates fonts against industry standards and generates reports highlighting issues that need attention. When conducting thorough testing, several environmental factors must be considered. Cross-platform testing should verify consistent rendering across Windows, Mac, Linux, and other operating systems, as well as various web browsers. Display variables are also important, requiring evaluation on different screen types, resolutions, and pixel densities. Font sizes should be tested across multiple ranges based on intended optical size—8 to 14 pt for body copy, 16 to 24 pt for subheadings, and 24 pt and above for headlines and titles. Additionally, fonts should be evaluated with various text rendering technologies, including DirectWrite, CoreText, and FreeType, to ensure consistent performance across different rendering engines.

The following are common problems that font designers run into:

## Collisions

Connections or overlaps of glyphs and diacritical marks where they should be separate.

- **Character overlaps:** Ensure characters don't collide.



Timelessness

Timelessness

Fig. 29 Collision in red circle (Latin) and its solution, done with kerning.

- **Diacritic positioning:** Ensure accents and other marks don't collide with base characters or each other.

ről lên bản in năm 1450  
ről lên bản in năm 1450

Fig. 30 A tone mark that often requires manual adjustment in red circle. (Latin)

- **Leading (line spacing) issues:** Check that ascenders and descenders don't collide with adjacent lines.

τη  
από → τη  
από

Fig. 31 Leading collision in red circle (Greek) in two separate lines of text. Potential solutions are adjusting vertical metrics, descenders, and combining marks.

- **Punctuation spacing:** Verify that quotes, parentheses, and other punctuation don't collide with other glyphs.

## Legibility

Legibility is the quality of being clear enough to read. This includes ensuring similar characters remain distinguishable in context. Consider how legible the font is for readers with visual impairments.

l liad      1 liad

Fig. 32 Some commonly confused characters in Latin: The numeral one, an uppercase l, a lowercase l, in two different fonts.

## Shaping

- **Mark attachment:** Validate that marks are positioned as expected relative to the base character.



Fig. 33 Mark attachment (N'ko). When a mark is not properly attached, it will either appear above a dotted circle next to the base character or appear offset from the base character.

- **Joining:** Test joining behavior in scripts like Arabic.

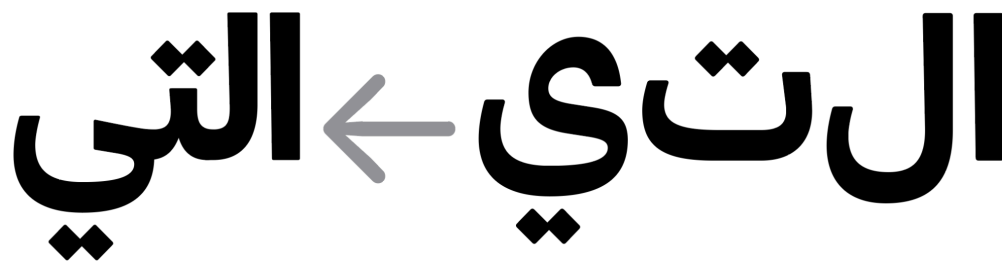


Fig. 34 Check that the features are working properly to support complex shaping. The individual characters (shown on the right) are replaced on the left with the appropriate positional forms to make connections on either side depending on each character's context and position. (Arabic)

## Advanced Topics

### Design Concepts

- **Black and White Density:** In many logographic and featural alphabet writing systems, a character is a cluster of various pictographic parts. Some letters have an intricate composition of these parts, and some are extremely simple. These characters cannot have the same percentage of form and [counterform](#), but are made to look as equal as possible by using optical compensation techniques.

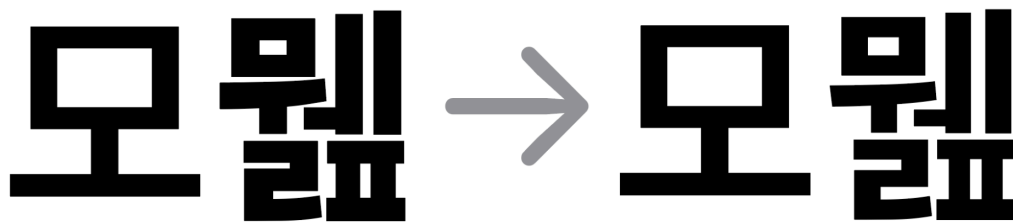


Fig. 35 A pair of Hangeul characters before optical compensation, and after. The more complex character on the right requires lighter stroke weights compared to the simpler character on the left, in order to balance the black and white density.

- **Stroke Weight Consistency:** The thickness of strokes (lines) used to form each glyph should follow consistent principles throughout your entire font to maintain visual harmony.



Fig. 36 Stroke weights (red arrows) in Latin and Arabic.

- **Stroke Contrast:** Varying the thickness of horizontal and vertical strokes to create visual balance, often making horizontal strokes thinner than vertical ones.



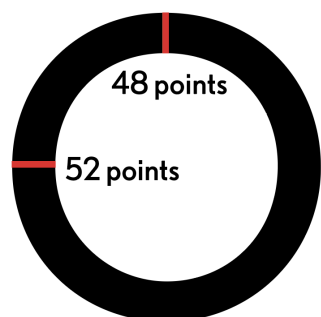


Fig. 37 An optically balanced circle. Horizontal strokes almost always have the illusion of higher thickness than vertical strokes of equal thickness.

- **Proportions:** forms and interior spaces should be related.



Fig. 38 On the left, the counterforms (blue) in D, P, R, and B vary in squareness, roundness, and inclination. They should have harmonious shapes, such as those on the right. (Latin)

- **Overshoots:** Round letters (like O, C, S) and pointed letters (like A, V) extend slightly above or below horizontal guidelines to appear visually aligned with flat letters.



Fig. 39 Overshooting a baseline. (Latin)

- **Crossbar Positioning:** Placing horizontal elements (like in "H" or "f") slightly above the mathematical center to appear centered to the human eye.

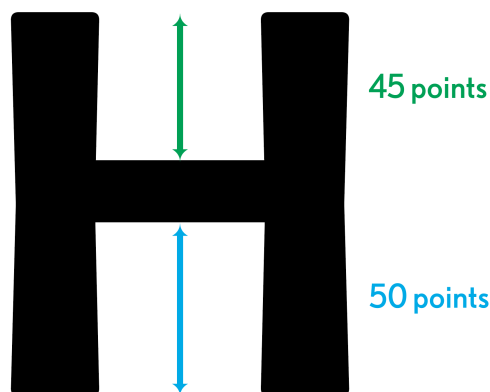


Fig. 40 Crossbar positioning. (Latin)

- **Diagonal Compensation:** Thickening diagonal strokes (like in "N" or "V") which would otherwise appear thinner than vertical strokes.

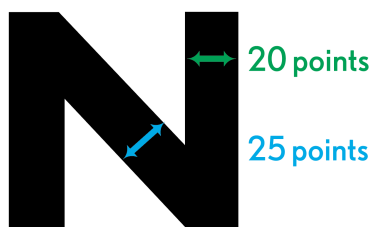


Fig. 41 Diagonal compensation in N. (Latin)

- **Weight Distribution:** Carefully distributing weight around curves to maintain consistent visual density.

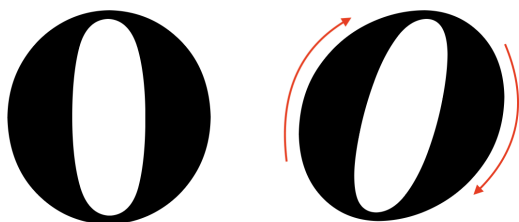


Fig. 42 Weight redistribution in upright O versus italic O (Latin, Greek, Cyrillic). In italics, the thickest part of the curve rotates up the left side of the "O" and down on the right side.

- **Ink Traps:** Small notches added to intersections that fill in during printing, preventing ink spread at small sizes.

# Nodes

Fig. 43 Ink traps in red circles. (Latin)

- **Optical Knots:** An intersection of strokes where there's a visible thickening of the lines in a letterform. These can turn into distracting dark spots that create irregular rhythm in a body of text.



Fig. 44 On the left, a B with an optical knot. On the right, the knot has been corrected. (Latin)

- **Serif/Terminal Balance:** Adjusting the thickness, length, and size of serifs or terminals based on their position to create visual harmony.



Fig. 45 Related serifs in descending sizes based on relative counterspaces. (Latin)

- **Kerning**  
Kerning (verb) is the act of making spacing adjustments between specific characters in a line of text. Kerning (noun) is the result of that act.



HAVANA

Fig. 46 The word HAVANA with no kerning. (Latin)

Once all the characters in your font are well-spaced, there may be exceptions that require kerning. For example: in Latin, a well-spaced “A” will look too far from a well-spaced V without kerning. Kerning will subtract (or add) values to a specific pair of letters to make them look consistent with the rhythm of the word they are in:



HAVANA

Fig. 47 The word HAVANA with appropriately balanced kerning. (Latin)

It is important not to overkern and to test kerning pairs in the context they would be used to make sure they are well-balanced in words:



HAVANA

Fig. 48 The word HAVANA with overkerning. (Latin)

## In Summary

This guideline provided you with a basic background for digitizing and visualizing text. With this information, you can assess the requirements that your font must satisfy for your language. You can then use font editing software as described to create your initial font. You may then solicit feedback from your language community, make improvements, and iterate, until the font is largely satisfactory. As you gain experience, you will realize there is more to learn, and the Glossary and Resources Appendices point you to helpful information and tools. Developing a font is often neither a simple nor a short-term project. However, it is very rewarding to ultimately see your language rendered appropriately for your culture, and to have your font used by your community and in numerous documents.

## Appendix: Digital Font Design Editors

Below is a table of some of the most commonly used professional font design editors. These were chosen for their popularity, effectiveness, price points, and ability to export web formats. They also all have trial versions and discount pricing. There are several more font design editors available; a more complete list can be found [here](#).

Editors	Glyphs	Font Creator	FontForge	FontLab	Fontra	RoboFont
Platform	Mac OS	Windows	MacOS	Mac OS/ Windows	Mac OS/ Windows	Mac OS
Price (as of 2025)	€49/299	\$49/149/199	Free	\$49/499	Free	\$490
Skill level	Easy	Easy	Hard	Moderate	Moderate	Moderate
Advanced Projects	Some	Yes	No	Yes	Yes	Yes
Right-to-left support	Yes	Yes	Yes	Yes	Yes	Yes
Vertical typesetting	Yes	Yes	Yes	No	Yes	Yes but slow
Light version	Glyphs Mini	Home	N/A	TypeTool	N/A	N/A

## Appendix: Glossary

Here are terms we've used in these guidelines. They are specific to typeface design; they may have other definitions in tangential fields. The sources for these definitions include: [©Words of Type Studio](#), [Glyphs Handbook](#) (© 2011–2024 Glyphs GmbH), and [RoboFont Documentation](#) (©TypeMyType).

**Anchor:** A tool within the user interface of the font design software that helps connect base characters with combining marks.

**Ascender:** In Latin, Greek, and Cyrillic, the part of the lowercase letter that ascends above the x-height. It can be above, below, or equal to the Cap height.

**Ascender (vertical metric):** A numerical value for the top of the bounding box.

**Autotracing:** a routine often found in illustration and font editing programs in which the program defines the edges of a scanned image with automatically generated vectors.

**Base character:** A basic character without any diacritics. Also known as a parent character.

**Baseline:** The main line on which all characters are positioned. Some go above, some below, depending on the writing system.

**Bounding box:** The rectangle that encloses a digital glyph. Its height and depth are set by vertical metrics.

**Cap height:** In Latin, Greek, and Cyrillic, the height of the “H”, which serves as a guideline for the height of all the other lowercase letters.

**Combining marks:** A specific kind of diacritic that is drawn to combine with base characters.

**Control character:** A character that has the basic building blocks of a writing system that is often drawn at the beginning of font creation to help determine the style and metrics of a font.

**Counters (or counterforms):** the space(s) inside a glyph that are defined by the shape of the glyph itself.

**Descender:** The part of a lowercase letter that descends below the baseline.

**Descender (vertical metric):** A numerical value for the bottom of the bounding box.

**Diacritic:** A glyph added to a letter indicating a difference in pronunciation or tone.

**Font binary file:** A file containing the instructions and data defining a font in a form optimized for computer programs to use to render and layout text. Example file extensions for a font binary file include: .OTF, .TTF, or .WOFF.

**Font source file:** A file containing the human-readable instructions and data to define a font. Example file extensions for a font source file include: .glyphs, .UFO, or .VFC. A program, for example OpenType, uses the source file contents to generate a font binary file. Computer programs use the binary file to implement the font and to render and layout text.

**Form:** Also known as positive space, this is the outline of the glyph.

**Glyph:** Any shape in a script or writing system that constitutes a letter, number, diacritic, tonal mark, number, punctuation mark, or other special symbol.

**Kerning:** (verb) The act of making spacing adjustments between specific characters in a line of text. (noun) The result of that act.

**Leading:** Also called “line spacing”, the distance between two lines of text. Often, that distance is measured between two baselines.

**OpenType features:** These are [features](#) that improve the display of the font. They can help with complex shaping. They can also help the user to make stylistic choices, such as character variations used by linguistic subgroups.

**Optical size:** Typefaces need to be optimized for specific sizes. A text typeface, meant to be read very small, should have more open counters and looser spacing. A display typeface, meant to be read very big, can have more delicate features and tighter spacing. Text typefaces look clunky at large sizes, and display typefaces are hard to read at small sizes.

**Parent character:** Another term for “base character.”

**Posture:** There are two main postures in Latin font design, Roman (upright) and italic (slanted).

**Set width:** the total width of a character, including the width of the glyph itself and the counterspace that surrounds it.

**Script:** (1) A style of fonts that mimics handwritten forms, (2) a writing system, or (3) a tool used to automate processes while creating a font.

**Side bearing:** The value of the horizontal space on the right or left side of a character, from the glyph to the end of the bounding box.

**Spacing:** Setting the values of a glyph’s side bearings, which influences the distance between each glyph combination. Spacing should be done at the same time as drawing in order to have a high-quality typeface.

**Style:** (1) a specific iteration of a typeface (e.g. Helvetica Condensed Thin Italic; Helvetica Bold), or (2) a classification of the overall look of the typeface (e.g. serif; sans).

**UFO file:** A Unified Font Object file, a cross-platform format for storing font data.



**Vertical metrics:** Vertical metrics determine the boundaries of a font as it displays on screen: the bounding box, the baseline relative to the bounding box, and the recommended gap between lines of text.

**X-height:** In Latin, Greek, and Cyrillic, the height of the “x”, which serves as a guideline for the height of all the other lowercase letters.

# Appendix: Resources

The following is a sampling of the most popular tools that professional typeface designers use.

## Tools

[Font Bakery](#): A web-based tool for checking the quality of font projects.

[FontSpector](#): A command-line tool for checking the quality of font projects.

[Font Goggles](#): An application for OpenType feature testing.

[Dutch Type Library OTMaster](#): An application for reviewing, editing, and altering tables and glyphs of fonts with an [sfnt-file](#) structure.

[fontTools and TTX](#): Command line tools for manipulating fonts in Python. TTX is especially helpful for converting fonts to XML text.

[Fontdev.app](#) A web-based tool for viewing and occasionally editing OpenType tables from a font binary file.

[FontTableViewer](#) An application for viewing and comparing OpenType tables of several binary files at once.

[Adobe InDesign](#): A layout application for creating test proofs.

[Canva](#): A popular web-based tool that can be used for font design testing.

[Figma](#): Another popular web-based tool that can be used font design testing

[Font Proofer](#): A plug-in tool for generating design proofs in Glyphs and RoboFont.

[Microsoft Word](#): A ubiquitous word processing application.

[LibreOffice](#): An open-source office productivity software suite.

[TextEdit](#): A basic word processing application on MacOS.

[Notepad](#): A simple text editor for Microsoft Windows.

[Pages](#): A basic layout application provided with MacOS.

## Further Resources

[Type Design Resources](#)

[Rosalie Wagner's Type Links](#)

[On Snot and Fonts](#)

[Words of Type](#)

[Glyphs Handbook](#)

[RoboFont Documentation](#)

[Aesthetic Innovation in Indigenous Typefaces: Designing a Lushootseed Font  
r12a.io](#)

[Glyphs Handbook](#)

[The Anatomy of Type](#)

[The Stroke: The theory of writing](#)

[Typeface Mechanics](#)

[OpenType Registered Features](#)

[Inside Paragraphs](#)

[Autotracing](#) (O'Reilly)

[Autotracing](#) (FontLab)

[Bi-scriptual](#)

# Acknowledgements

This guideline is developed through Translation Commons' Language Digitization Initiative — an effort aligned with UNESCO's International Decade of Indigenous Languages. These resources aim to empower Indigenous and minority language communities to bring their languages online.

If your community would like assistance with digitizing your language, contact [krista@translationcommons.org](mailto:krista@translationcommons.org) or visit [TranslationCommons.org](https://TranslationCommons.org).

Translation Commons thanks its many volunteers who contributed their expertise and significant time to produce this guideline.

The following trademarks are mentioned: Glyphs, FontLab®, TypeTool™, FontCreator, RoboFont, Fontra.

## **Version 1.0 - 2025**

*Lead Authors: Mark Jamra, Neil Patel, Tex Texin, Dyana Weissman*

*Primary Editors: Dyana Weissman, Kathryn Hoesly*

*Reviewing Editor: Julie Anderson*

*Contributors: Sahar Afshar, Min-Young Kim, Dev Kumar*