Babel

v0.1.1 MIT-0

Redact text by replacing it with random characters

Maja Abramski-Kronenberg <HTTPS://ME.DIGITALWORDS.NET>

https://codeberg.org/afiaith/babel

This package provides functions that replace actual text with random characters, which is useful for redacting confidential information or sharing the design and structure of an existing document without disclosing the content itself. A variety of ready-made sets of characters for replacement are available (75 in total), representing diverse writing systems, codes, notations and symbols. Some of these are more conservative (such as emulating redaction using a wide black pen) and many are more whimsical, as demonstrated by the following example:

```
#baffle(alphabet: "welsh")[Hello]. My #tippex[name] is #baffle(alphabet: "underscore")[Inigo Montoya]. You #baffle(alphabet: "alchemy")[killed] my #baffle(alphabet: "shavian")[father]. Prepare to #redact[die].

Using show rules strings, regular expressions and other selectors can be redacted automatically:

#show "jan Maja": baffle.with(alphabet: "sitelen-pona")
#show regex("[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*"): baffle.with(alphabet: "maze-3")

I'm jan Maja, and my email is `foo@digitalwords.net`.
```

Lliwyn. My   is ____ ______. You 🜪🜨🜋🝆sss🜵🜠 my ·𐑡𐑝𐑮𐑛𐑜𐑴. Prepare to [redacted].

Using show rules strings, regular expressions and other selectors can be redacted automatically:

I'm [illegible], and my email is [illegible]@[illegible].[illegible].

Part I.
Introduction

I.1. Purpose and usage scenarios

At times one wishes to make portions of text (or the whole text) hidden from the recipient:

- The most common case is when redacting confidential information. Traditionally this is done by overwriting portions of text with a wide black pen and photocopying the result.
- Another usage scenario is sharing the design and structure of a document, but not the text itself. While *lorem ipsum*[1] blocks help when demonstrating the design of a template — replacing places where actual text would go with placeholder text — when sharing the way a particular existing document looks they are less helpful, since in order to use them one would have to make a copy of the document and manually substitute text with placeholder text of more or less the same length, which is tiresome and prone to errors.

In addition, playing with various contemporary, historical and constructed writing systems is a special kind of geeky fun... While the package does have serious, practical use, most of the provided alphabets (§ II.2) are there just for fun.

One thing I ask you to avoid is using Babel for mocking cultures, as often done with mimicry typefaces such as Faux Cyrillic°, Faux Hebrew° or Wonton font°, which have more than subtle racist undertone. This package is a celebration of the variety and diversity of writing[2].

If you wish to share the Typst source files of your document, not just the precompiled output, a tool called *Typst Mutilate*° might be useful for you. Unlike Babel, it is not a Typst package but an external tool, written in Rust. It replaces the content of a Typst document with random words selected from a wordlist or random characters (similarly to Babel), changing the document in place (so make sure to run it on a *copy*!). As a package for Typst, Babel cannot change your source files.

I.2. Name

Have a seat, it's story time. Babel is named so as a wordplay on two things: the Biblical myth of the Tower of Babel and the LaTeX package° sharing the same name. For anyone who isn't familiar with the story, here is the full fragment (Genesis 11.1–9):

וַיְהִי כָל־הָאָרֶץ שָׂפָה אֶחָת וּדְבָרִים אֲחָדִים׃ [1]
וַיְהִי בְּנָסְעָם מִקֶּדֶם וַיִּמְצְאוּ בִקְעָה בְּאֶרֶץ [2]
שִׁנְעָר וַיֵּשְׁבוּ שָׁם׃ [3] וַיֹּאמְרוּ אִישׁ אֶל־רֵעֵהוּ
הָבָה נִלְבְּנָה לְבֵנִים וְנִשְׂרְפָה לִשְׂרֵפָה וַתְּהִי
לָהֶם הַלְּבֵנָה לְאָבֶן וְהַחֵמָר הָיָה לָהֶם לַחֹמֶר׃
וַיֹּאמְרוּ הָבָה | נִבְנֶה־לָּנוּ עִיר וּמִגְדָּל וְרֹאשׁוֹ [4]

[1] Now all the earth was of one language and one set-of-words. [2] And it was when they migrated to the east that they found a valley in the land of Shin'ar and settled there. [3] They said, each one to his neighbor: Come-now! Let us bake bricks and let us burn them well-burnt! —;For them brick-stone was like building-stone, and raw-bitumen was for them like red-mortar. [4] And they said: Come-now! Let us build

[1] Typst provides a built-in function for this, `lorem()`°, which outputs pseudo-Latin. For a Japanese blind text generator, see Roremu°.

[2] Just look at § II.2! Human beings — as well as Klingons and Elves... — came up with so many different graphic symbols to represent sounds and ideas it's mind-boggling.

ourselves a city and a tower, its top in the heavens, and let us make ourselves a name, lest we be scattered over the face of all the earth! [5] But YHWH came down to look over the city and the tower that the humans were building. [6] YHWH said: Here, [they are] one people with one language for them all, and this is [merely] the first of their doings — now there will be no barrier for them in all that they scheme to do! [7] Come-now! Let us go down and there let us baffle their language, so that no one will understand the language of his neighbor. [8] So YHWH scattered them from there over the face of all the earth, and they had to stop building the city. [9] Therefore its name was called Bavel/Babble, for there YHWH baffled the language of all the earth-folk, and from there, YHWH scattered them over the face of all the earth.

בַּשָּׁמַיִם וְנַעֲשֶׂה־לָּנוּ שֵׁם פֶּן־נָפוּץ עַל־פְּנֵי כָל־הָאָרֶץ׃ [5] וַיֵּרֶד יְהוָה לִרְאֹת אֶת־הָעִיר וְאֶת־הַמִּגְדָּל אֲשֶׁר בָּנוּ בְּנֵי הָאָדָם׃ [6] וַיֹּאמֶר יְהוָה הֵן עַם אֶחָד וְשָׂפָה אַחַת לְכֻלָּם וְזֶה הַחִלָּם לַעֲשׂוֹת וְעַתָּה לֹא־יִבָּצֵר מֵהֶם כֹּל אֲשֶׁר יָזְמוּ לַעֲשׂוֹת׃ [7] הָבָה נֵרְדָה וְנָבְלָה שָּׁם שְׂפָתָם אֲשֶׁר לֹא יִשְׁמְעוּ אִישׁ שְׂפַת רֵעֵהוּ׃ [8] וַיָּפֶץ יְהוָה אֹתָם מִשָּׁם עַל־פְּנֵי כָל־הָאָרֶץ וַיַּחְדְּלוּ לִבְנֹת הָעִיר׃ [9] עַל־כֵּן קָרָא שְׁמָהּ בָּבֶל כִּי־שָׁם בָּלַל יְהוָה שְׂפַת כָּל־הָאָרֶץ וּמִשָּׁם הֱפִיצָם יְהוָה עַל־פְּנֵי כָּל־הָאָרֶץ׃

The myth explains why people are scattered everywhere and why they speak different languages, and it also provides folk etymology for the name of the city of Babylon (בָּבֶל *Bāḇel*): בָּלַל *bālal* ‘he mixed, he confounded’ in verse 9 and וְנָבְלָה *wə-nāḇlâ* ‘and let us mix, and let us confound’ in verse 7 (both from the root ב־ל־ל *√BLL* ‘to mix, to confound, to confuse’) sounds a bit like בָּבֶל *Bāḇel* ‘Babel’.[3] Everett Fox translated these verbs brilliantly in the *Schocken Bible*°, with the English verb *baffle*, where all other translation I looked at have *confound* or *confuse*. This is the reason for choosing that translation for the above excerpt and the name `#baffle()` for the main function provided by Babel.

Now, idea of the Tower of Babel as the explanatory myth behind linguistic diversity still persists in contemporary culture, as demonstrated by the Babel fish in Douglas Adams’s *the Hitchhiker’s Guide to the Galaxy*, the dictionary and machine-translation software *Babylon* and the LaTeX package *Babel*°. Fittingly, the Babel LaTeX package enhances the capabilities of localisation and internationalisation. With our Typst Babel I chose to take the other connotation, of confusion, bafflement and mixing 🙃

I.3. Logo

The logo features a minimalist icon of the Tower of Babel or a ziggurat; see § I.4 for attribution. The background colour is the same shade of turquoise used by Mantys.

I.4. Copyright and licence

The this package is released under MIT-0°.

Babel’s logo features an image° by Andrejs Kirma° which is released under CC BY-3.0. I attribute them willingly, as I find the graphics very fitting for the logo. Go check their other icons°.

[3] Interestingly, the Babylonian Akkadian name which is the basis for the Hebrew name is 𒆍𒀭𒊏𒆠 *Bābilim* ‘(lit.) the gate of the gods’. Even more interestingly, there is evidence this Akkadian interpretation of the name (as ‘the gate of the gods’) itself was a Semitic folk etymology on a non-Semitic name! This is all very… confusing, how people mix up things.

I.5. Versioning and stability

BABEL follows the Semantic Versioning scheme (SemVer 2.0.0°). While it is fully usable in its current form (version `0.*.*`), changes to the API might occur in future versions. This should not pose a problem:

- When you import a package in Typst you can indicate the version (for example, `#import "@preview/example:0.1.0"`), so no surprises should occur.
- Changes to the API will be clearly indicated in the documentation.[4]

I.6. Participation and contact

If there is anything that doesn't work well or any feature you want added or changed, don't hesitate to open an issue° on the Git repository, and I will do my best to make the package more useful for you and others. If you want to contribute code/documentation (changes, additions, corrections, improvements, etc. no matter how small or large), pull requests° are very welcome; thanks!

In particular, contributions of alphabets (`src/alphabets.yaml`) are welcome. This version contains 75 (!) alphabets, but like Pokémon, you gotta catch 'em all... When choosing a font for the script of your alphabet:

- If only basic Latin characters are used, don't set a font.
- If the new alphabet uses a script already represented on BABEL, prefer the font already in use (for example, Gentium Plus for Latin, Greek and Cyrillic, SBL Hebrew for Hebrew, ...).
- Prefer free, gratis and libre open-source fonts (FLOSS).
- Prefer serif[5] fonts (wherever serif makes sense) that go well with Gentium Plus.

Not everyone is familiar with Git, so if that is a problem feel free to contact me in any other way; see `https://me.digitalwords.net/`° for contact information.

I.7. Disclaimer

I hold no responsibility for anything that may occur as a result of using this package, nor can I guarantee there are no edge cases where text that should have been redacted stays readable (please do report such cases; see § I.6). If you use this package with actual confidential information, please read the manual (especially § II.1.2), check the results and understand the risks.

[4] If the characters replaced by the package change between versions, this is not counted as a change: the whole point is the actual identity of the random characters is, well, random. Changes to the alphabets (§ II.2) are also considered minor.

[5] Admittedly, for many scripts Noto Sans is the only good FLOSS option.

Part II.
Usage

II.1. Provided functions

```
#baffle(
  ⟨alphabet⟩: "latin-bicameral",
  ⟨case-sensitive⟩: true,
  ⟨punctuate⟩: true,
  ⟨punctuation⟩: "punctuation",
  ⟨input-word-dividers⟩: "(\" \",)",
  ⟨output-word-divider⟩: " ",
  ⟨set-font⟩: true,
  ⟨seed⟩: none,
  ⟨as-string⟩: false
)[⟨input⟩] → content°
```

> ⚠ A technical note: the default of ⟨punctuation⟩ is `"\"\\`\\-=~!@#$%^&*()_+\\[\\]\\\\;':\",./<>?‘’“”\u{f8fd}\u{f8fe}·¿¡ʕ,lll◊ºظ\""` and that of ⟨input-word-dividers⟩ is `(" ",)` (that is, an array containing only a space, U+0020); for reasons that have to do with MANTYS or TIDY the default values are displayed wrongly in this document as `"punctuation"` and `"(\" \",)"` respectively.

Replaces ⟨input⟩ with random characters chosen from a given ⟨alphabet⟩. Note that depending on the ⟨alphabet⟩ the output might be longer then the input (in case a single letter is replace by a digraph, for example).

Argument

⟨input⟩ — content°

The text to be redacted.

```
#baffle[A _confidential_ text]
```

U *quijntrvhyfcx* ebap

Argument

⟨alphabet⟩: "latin-bicameral" — str° | dictionary°

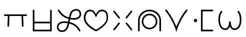
Either a slug (a string identifier) referring to the alphabet to be used for the output (options are listed in § II.2) or a dictionary describing the alphabet, with the following fields:

lowercase *(required)* an array of strings (each representing one character), from which a random string is drawn for each character in the original text. If the target script is bicameral, use this field for lowercase letters and the uppercase field for uppercase ones; otherwise, use only this field.

`uppercase` *(optional)* an array of the same size as `lowercase`; if the input is bicameral, uppercase letters in the input are matched by uppercase letters from the alphabet.

`font` *(optional)* the font to typeset the output in.

```
#baffle(alphabet: "sitelen-pona")[top secret]\
#baffle(alphabet: "greek")[top secret]\
#baffle(alphabet: (lowercase: ("👍", "👎")))[top secret]
```



αζτζ αχτβυιμπ

👍👎👎 👎👍👎👍👎👍

Argument

`⟨case-sensitive⟩: true` `bool°`

Indicates whether to retain (`true`) case sensitivity or ignore (`false`) it, making everything lowercase.

```
#baffle[Hello] vs. #baffle(case-sensitive: false)[Hello]
```

Iguau vs. xauie

Argument

`⟨punctuate⟩: true` `bool°`

Indicates whether to retain (`true`) punctuation or ignore (`false`) it, treating punctuation marks like regular letters.

```
#baffle[hello!] vs. #baffle(punctuate: false)[hello!]
```

uulqua! vs. uadokr

Argument

`⟨punctuation⟩: "punctuation"` `str°`

A string containing all characters considered a punctuation mark.

Argument

`⟨input-word-dividers⟩: "(\" \",)"` `array°`

The set of characters considered word dividers in the input, to be replaced by `⟨output-word-divider⟩` in the output.

```
#baffle[hello·world] vs. #baffle(input-word-dividers: ("·",))[hello·world]
```

---

poeui·yzlque vs. onapy gfjti

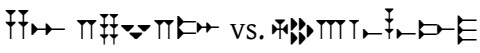
For writing systems that have spaces between words, leave as it is, but if your input text is Tibetan for example, `("\u{0f0b}", " ")` (*tsek* and space) might be a better option, otherwise you'd get *very* long words in the output.

Argument

⟨output-word-divider⟩: " " str°

A string to which any character in ⟨input-word-dividers⟩ is converted.

```
#baffle(alphabet: "ugaritic")[два слова] vs. #baffle(alphabet: "ugaritic", output-word-divider: "𐎗")[два слова]
```

 vs.

Keep " " if your target alphabet uses spaces; change to `"\u{200b}"` (zero-width space) if it doesn't and there is no special word-dividing symbol such is in Ugaritic (zero-width space allows line breaking, whereas an empty string, `""`, does not).

Argument

⟨set-font⟩: true bool°

Indicates whether to typeset the output in the surrounding font (`false`) or the font suggested by BABEL (`true`).

Argument

⟨seed⟩: none int° | none

If provided (not `none`), used for initialising the random number generator with that seed.

```
#baffle(seed: 42)[hello] vs. #baffle(seed: 1312)[hello] vs. #baffle(seed: 42)[hello]
```

---

uixzf vs. ibaih vs. uixzf

Argument

⟨as-string⟩: false bool°

Treat the input as a *string* (as opposed to *content*). This argument exists because of technical limitations of Typst (at least as of version 0.11.0), where some abilities cannot coexist:

Feature	false	true
Formatting and complex text capabilities	○	✕
Spaces between formatting groups (see this issue°)	✕	N/A
Scripts with contextual letter forms	✕	○
Counting of characters	codepoints	graphemes
Changes the table of contents; see § II.1.2.2	✕	○

The choice between the two modes depends on what you need, and in some cases compromises cannot be avoided; for example, at the moment it's not possible to apply `#baffle()` on a heterogeneous span of formatted Arabic or Devanāgarī text, unless you manually surround each individual homogeneous formatting group with a `#baffle()` command where ⟨as-string⟩ is set to `true`.

```
#table(
columns: 3,
table.header([Feature], [#arg("as-string")`: `#value(false)], [#arg("as-string")`:
`#value(true)]),
table.hline(stroke: 0.05em),
[Formatting and spaces],
ylwcell(baffle(output-word-divider:"@")[one two *three*]),
ylwcell(baffle(output-word-divider:"@", as-string: true)[one two *three*]),
[Contextual forms],
redcell(baffle(alphabet: "arabic")[hello]),
grncell(baffle(alphabet: "arabic", as-string: true)[hello]),
[Precomposed _â_ (`U+00E2`)],
grncell[#baffle(alphabet: "alchemy")[â] (1)],
grncell[#baffle(alphabet: "alchemy", as-string: true)[â] (1)],
[Combining _â_ (`U+0061 U+0302`)],
redcell[#baffle(alphabet: "alchemy")[â] (2)],
grncell[#baffle(alphabet: "alchemy", as-string: true)[â] (1)],
[Multi-codepoint\ emoji 🔄 (`U+1f1E6 U+1f1F6`)],
redcell[#baffle(alphabet: "alchemy")[🔄] (2)],
grncell[#baffle(alphabet: "alchemy", as-string: true)[🔄] (1)],
)
```

Feature	⟨as-string⟩: false	⟨as-string⟩: true
Formatting and spaces	eae@jms **oxays**	znl@mxu@ijntr
Contextual forms	ش‌ت‌خ‌ج‌ش	ص‌ا‌م‌ذ
Precomposed â (U+00E2)	𝔛 (1)	⊙ (1)
Combining â (U+0061 U+0302)	⊕⊜ (2)	◇ (1)
Multi-codepoint emoji 🇦🇶 (U+1f1E6 U+1f1F6)	◇◻ (2)	T (1)

#redact

A synonym of `#baffle()` with ⟨alphabet⟩ set to `"redaction"`.

```
This is #redact[confidential].
```

This is [redacted].

#tippex

A synonym of #baffle() with ⟨alphabet⟩ set to "tippex".

```
This is #tippex[confidential].
```

This is [tippexed].

If you frequently use #baffle() with certain parameters, defining an alias of your own makes things simpler, easier, and more elegant; for example:

```
#let tp = baffle.with(alphabet: "sitelen-pona", punctuate: false, output-word-divider:
"\u{200b}")
Hi! #tp[this!] and #tp[that...] are confidential.
```

Hi! [baffled] and [baffled] are confidential.

II.1.1. Using show rules

While surrounding segments of commands is useful for short amounts of text, applying commands to long segments — or even the whole document — is cumbersome. Fortunately, Typst provides us with a clever solution for that: show rules. Consider the following example:

```
This text is shown as plaintext.

#show: baffle.with(alphabet: "astrology")

Now from here on the text is baffled!
```

This text is shown as plaintext.

[baffled]!

Show rules can be used for redacting strings, regular expressions and other selectors automatically (see the documentation°):

```
#show "Ramona Flowers": baffle.with(alphabet: "redaction")
#show            regex("[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*"):
baffle.with(alphabet: "maze-3")

Her name is Ramona Flowers, and her email is `ramona@flowers.name`.
```

Her name is [redacted], and her email is [redacted].

At the moment (version 0.11.0) there is no way to revoke show rules[6]. As a workaround, bracketing the relevant segments can limit the scope of a show rule:

```
This is plaintext.

#[
  #show: baffle

  A baffled part of the document.
]

Plaintext again.
```

This is plaintext.

I ynaazli jbyu ig diy oatsenhe.

Plaintext again.

II.1.2. Limitations and considerations

II.1.2.1. General matters

II.1.2.1.1. Educated guesses

From an information-theoretical point of view, the `#baffle()` loses a lot of information: it may retain only the length of each word, capitalisation, and punctuation, and each of the three can be turned off. Unlike encryption, where by definition the plaintext can be deciphered using the proper technique and secret information, here the original text cannot be recovered algorithmically. Given certain conditions, educated guesses can be made though: for example (and an extreme one at that…), if you see a baffled word that is 27 letters long, you can assume with confidence that it is *electroencephalographically* if the text deals with neurology or *ethylenediaminetetraacetate* if it deals with chemistry. In more realistic scenarios contextual information and the nature of the text (rigid forms are more predictable than literary prose, for example) can assist in making educated guesses.

For maximal obfuscation, turn word division, capitalisation and punctuation off, but realistically I don't think this is needed, and the result might be less aesthetically pleasing, depending on the alphabet used.

[6] It is planned, though. See the roadmap° and this issue°.

Note that `#baffle()` retain information about the length of words in characters[7], not in horizontal length. This covers some attack vectors involving proportional fonts and kerning, but opens others, based on counting characters.

II.1.2.1.2. Unintended meanings

Letters are chosen from the alphabets randomly. In theory, unintended meanings may occur, especially with relatively short words. Also, note that some scripts (Egyptian hieroglyphs and Phaistos Disc, sitelen sitelen, …) contain pictograms which might be taboo in your culture.

II.1.2.2. Particular, implementation-dependant matters

Limitations concerning the ⟨as-string⟩ argument are discussed in § II.1, and those concerning particular alphabets are discussed in § II.2.

If elements which are included in the table of contents appear within `#baffle()`, they appear in plaintext in the PDF table of contents[8] unless they are bound within a `#baffle()` command with ⟨as-string⟩ set to `true`, in which case the bound text disappears from the PDF table of contents. The baffled text in a typeset table of contents (`#outline()`) is different to the one used in the headings.

II.2. Provided alphabets

In total 75 alphabets are provided by BABEL. ‘Alphabet’ is used here in the sense used in formal language theory, not linguistics (neither in the narrow nor the wide sense).[9] Many of the ‘alphabets’ below are not alphabets in the linguistic sense.

II.2.1. Legend

W	a link to the relevant article in Wikipedia. If anything piques your interest, down to the rabbit hole you go.
Aa	a link to the font used in the example.
`slug`	the string you provide the ⟨alphabet⟩ argument with (`arabic` for Arabic, `alchemy` for Alchemical symbols, …).

II.2.2. Notes

- The characters from the output alphabet are chosen at random, which has several implications:
  - Phonotactics is not taken into consideration.

[7] Some alphabets may increase the number of characters, as they include multi-character ‘letters’ (e.g. digraphs).

[8] This is the one accessible using a sidebar or the tab key, depending on the PDF reader, not the one typeset in the document itself.

[9] If these distinctions are not clear to you, read the following Wikipedia articles if you’d like to learn about them: Alphabet° (linguistics, both senses) and Alphabet (formal languages)°.

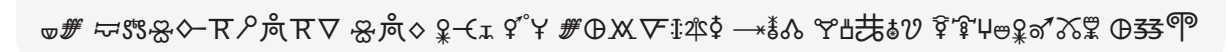
- The output contains mostly non-words which defy the rules of how words look in the alphabet in question.
- Letter frequency is also not taken into consideration. At most, the vowels or consonants are superficially doubled in order to account for severe disparity in the type:token ratio.

Because characters are chosen at random final letters have been removed from the Hebrew script and the Canadian Aboriginal syllabics, so they will not occur in incorrect positions.[10]

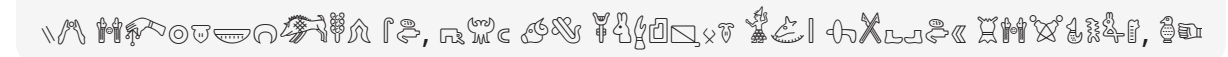
- Some of the scripts — such as Egyptian and Anatolian hieroglyphs or the sitelen pona and sitelen sitelen scripts — are normally written with grouping of characters in a non-linear manner. This is not done here, where the glyphs are stringed one after the other in a linear manner.

II.2.3. A menu of alphabets

Alchemical symbols `alchemy` W° Aa°



Anatolian hieroglyphs `anatolian` W° Aa°

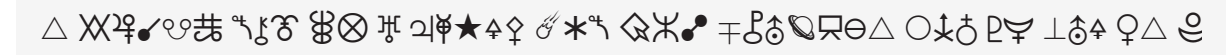


Arabic كتابة عربية `arabic` W° Aa°

ضطشتس كبص قض كضحتسرغط هرظودقز عرمط وظ ضهر كحغ فذ جعبثعهيز خل وز اصقض زلهزث زسج تبد خثغخز جقثخهكدر زر هعكرشذ

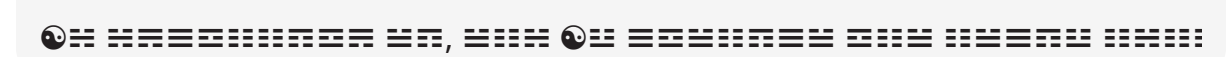
Only the letters of the Arabic script used for writing Arabic are included (not Persian, Urdu, Ottoman Turkish and other alphabets that use the Arabic script).

Astronomical and astrological symbols `astrology` W° Aa°



This alphabet includes all unicode characters from the Wikipedia articles about astronomial° and astrological symbols°. Some of the symbols are shared by `alchemy`.

Bagua `bagua` W° Aa°



[10] It's a better compromise not to represent the final forms than to have them occur in incorrect positions.

Baybayin ᜊᜌᜒᜊᜌᜒᜈ᜔ baybayin W° *Aa°*

ᜈ᜔ ᜇᜒᜇᜓᜎᜒᜐᜒᜊᜒᜊᜌᜒᜈ᜔ ᜒᜎᜈ᜔, ᜎᜓᜏᜓ ᜒᜎ ᜈ᜔ᜈᜓᜎᜐᜒᜈ᜔ ᜒᜊᜒᜎ ᜒᜊᜒᜈᜒᜇᜇ ᜒᜒᜒᜎᜓᜇᜒᜇᜈ᜔, ᜒᜓᜎᜓ ᜑᜒ

Brāhmī 𑀩𑁆𑀭𑀸𑀳𑁆𑀫𑀻 brahmi W° *Aa°*

𑀘𑀸 𑀘𑁂𑀵𑀻𑀺𑀺𑀢𑀩𑁆𑀭𑀸𑀳𑁆𑀫𑀻 𑀓𑀼𑁂𑀶𑁃 𑀮𑀯𑀺𑀦𑀸 𑀑𑀇𑀉𑀊𑀏𑀐𑀑𑀒 𑀅𑀍𑀬 𑀰𑀴𑀷𑀹𑀇 𑀜𑀲𑀑𑁆𑀓𑁆𑀶𑁄𑀪𑀬𑁆 𑀦𑁄𑀸 𑀘𑀸 𑀓𑀺

Braille ⠃⠗⠁⠊⠇⠇⠑ braille W° *Aa°*

⣭⣿ ⠻⣷ ⠹⣷⠄⠹⣷⠆⣿ ⠻⣷, ⠻⣷⠂⠂ ⠹⣿ ⣿⣿⣷ ⠂⣿ ⠻⣷⠄ ⣷ ⠂⠂⣿⣷ ⠹⣷, ⠹⣷⣿ ⣿⣷ ⠻⠆⠂ ⠂⠂ ⣿⠻⣿⣿

Canadian Aboriginal syllabics canadian W° *Aa°*

ᗴᗧ ·<·ᒪᕿᑕᕽᓄᓯᖁᘐ ᐷᑐ, ᓕᑕᐅ ·ᕒᔆᔭ ᗒ·ᔦᕿᖇᖘᔆᐸᑐ ᐁ·ᓀ·ᔨ ᑕᖁᓕ·ᘻᖮ ᑎᐷᐅᐁᖁᐷᖇᘐ, ᓚᕒᘏ ᗰᗷ ᐸᕮᖇ ·ᔦ

Cherokee syllabary ᏣᎳᎩ cherokee W° *Aa°*

HᎤ ᏯSᎤᏧBEᏲGᏯ ᏣᏫ, ᏫᏎᏫ SJ ᎮᏒAΓᏬᏗi ᏣPᎵ ᎩᏎEᎤᏫ ᏯIᎩOIHSᏔ

Chess pieces chess W° *Aa°*

♘♕ ♘♕♕♕♖♗♗♕♗ ♘♟, ♕♔♟ ♚♜ ♕♔♜♕♗♜♜ ♙♟♗ ♖♘♔♕♗ ♘♟♔♕♗♜♜♘♟

Cirth ᚴᛁᛕᛏ cirth W° *Aa°*

ᛔᛞ·ᚺᛉᚦᛊᛚᚱᚴᚠᛔᚦ·ᚲᛇᚴ·ᛗᚴᚷᚱᚴ·ᛏ·ᛜᚹᛉᚹ~ᚷᛟ·ᛘᛔᛉ·ᚳᚷᛔᚹᚺᛕᚴ·ᚣᚠᚳ·ᛚᚢᚾᛏᚹᛝ·ᚳᛉ·ᛋᛇᛟᚨ

Coptic ⲧⲙⲛ̄ⲧⲣⲙ̄ⲛ̄ⲕⲛⲙⲉ coptic W° *Aa°*

ⲁⲱ ⲧⲍⲃⲱⲣⲩⲟⲩⲥⲟ ⲁⲩ, ⲟⲓⲍ ⲭⲡ ⲁⲭⲁⲩⲟⲩⲅⲫ ⲫⲍⲕ ⲟⲩⲥⲙⲉⲉ ⲡⲛⲛⲩⲭⲩⲟ, ⲣⲍⲭ ⲥⲥ ⲉⲟⲩⲍ ⲑⲥ ⲁⲉⲓⲟⲩⲩⲫⲧⲓⲉ

Coptic (including uppercase) ⲧⲙⲛ̄ⲧⲣⲙ̄ⲛ̄ⲕⲛⲙⲉ coptic-with-uppercase W° *Aa°*

Ⲣⲟⲩ ⲩⲡⲁⲡⲩⲧⲱⲟⲍ ⲫⲱ, ⲥⲉⲓⲉ Ⲡⲩ ⲗⲭⲣⲫⲛⲩⲣ ⲩⲇⲣ ⲟⲣⲱⲧⲏ ⲣⲍⲩⲱⲣⲱⲕ, ⲫⲧⲟ ⲧⲍ ⲟⲩⲧⲡ ⲗⲱ ⲑⲉⲓⲍⲉⲛⲟⲣ

This variant includes uppercase forms. Note that the way they are used in Coptic is different from English and most languages written in the Latin alphabet (and modern Greek for that matter).

Devanāgari देवनागरी `devanagari` W◦ *Aa*◦

थेजि घीमोवाजंठूकोकयूटौ नृझ, वूमीण् तत् दिगैचिओझैजाढि फिघिज् टृझूङ्खोढ् खिचोज्चेनैफोतौ, फैटेङो ठाल

Nucleic acid notation (DNA) `dna` W◦

AT ATTCACATC AC, CAT AT GCCATAC AGG TAGGG ACAGCGA, TCC CA CCC CA TGGTGCCT GC

D'ni `dni` W◦ *Aa*◦

[illegible]

Dodecaphony `dodecaphony` W◦

DF♯ B♭EBECDD♯FC BF, EEC♯ FG♯ G♯A♯AEFD♭D♭ C♯DF D♭FBDG A♭CEFB♭CG, C♯DD E♭A♭ C♯G♭D

Egyptian hieroglyphs [illegible] `egyptian` W◦ *Aa*◦

[illegible]

Leiden Unified transliteration of Ancient Egyptian `egyptian-leiden` W◦ *Aa*◦

ts ẖtkbïẖqbï dss ḥmr ꜥḏ ïṯmmḏhï ïẖb tbnrr ꜥyḏsppsm fdg zt dꜣq ïz bṯstzrrt kh ṯtrzqqr sf ktï

Emoji `emoji` W◦

[illegible]

English English `english` W◦

My oyfbxgcxgch ct, diegh Fue hrkcieushau vyoy hytggh eashpffeiwhoo, uiquew ues aiiesc

Esperanto Esperanto `esperanto` W◦ *Aa*◦

Ij bdljeŭugkv ŝh, ŝheŭ Oaŭ kuĥoĉhe uoi adcĝĉ tgipĉoz, ovj ua miu je fueŭuagaŭd al obŭaoc

Elder Futhark ᚠᚢᚦᚨᚱᚲ `futhark-elder` W◦ *Aa*◦

ᚢᚾᛟᛊᛊᛃᛇᚲᛁᛁᛉᛊᚢᚱᛗᛈᚱᛞᚱᛇᛗᛉᛉᛁᛗᚱᛒᛗᚲᚠᚺᚺᛁᛗᛃᚦᚠᚱᚾᚠᚺᛗᛃᛗᛁᚾᛏᛁᚷᛚᛏᚹᛒᛚᛊᚢᛚᚦᛏᛞᛇᚢᛁᛞᚲᚱᛒᛚᛗᛊᛊᛚᚢᛟᛗᛏᛃ

Younger Futhark (long-branch) ᚠᚢᚦᚬᚱᚴ `futhark-younger-long-branch` W◦ *Aa*◦

ᛁᚴᚽᛏᛋᚬᚴᚽᚴᛋᚴᚦᚴᚽᚠᛘᚠᚼᛘᛘᚼᚴᚱᚠᛏᚠᚴᚦᛘᚢᚽᛦᛋᛦᚠᚱᛘᛏᛏᚬᛒᚴᛁᚬᚬᚽᚽᚦᛦᚱᚼᚴᛦᛒᛦᚼᛏᚴᚠᚴᚬᚬᛒᛋᛒᚴᚽᚬᛋᚽᚽᛒᛒ

Younger Futhark (short-twig) ᚠᚢᚦᚭᚱᚴ `futhark-younger-short-twig` W◦ *Aa*◦

ᚱᚴᛙᛙᚴᛁᚭᛁᛌᚭᚽᛏᛏᚠᛏᛏᚭᚭᛌᚠᚦᛙᚱᛌᚴᚢᚽᛏᛌᚽᛌᛌᚴᚦᚢᛁᚴᚭᚽᛌᛏᚴᛙᚦᚽᛂᚱᚴᚢᛙᛙᚴᚴᛁᛌᛙᛌᚱᚠᚦᚴᛌᛙᛙᚭᚢᚠᚴᚽᚢᛂᛏᚠᚦᚱᛌᚠᛙᛁᚭᚽᚱᚦᚢᚴᚢᛙᚭᛂᛏᚽᚽᚢᚽᚽᚭᛂᛁᛌ

Futhorc ᚠᚢᚦᚩᚱᚳ `futhorc` W◦ *Aa*◦

ᚳᛟᚩᛋᛄᛄᛞᚻᚳᚩᛄᚱᚱᚾᚦᚳᚩᚳᚷᛚᚴᚪᛖᚫᚹᚪᛝᛁᛞᛗᛚᛟᛚᚴᛈᛈᛄᛝᚩᚻᛝᛞᛋᚷᛗᛋᚩᚩᛖᛒᚾᚴᚴᛗᛏᛉᚹᛁᚪᚩᛋᛞᛇᛄᚹᛞᛠᛚᛞᛟᚴᚻᚲ

Geʿez *ግዕዝ* `geez` W◦ *Aa*◦

ኪኢ ካጱኢፐያሄጨጧጐ ጐር, ሑሶኮ ኢክ ሎስሀሪገሡዬ ጕገኢ ፏጸሓዜሣ ሒቲፏጰንማኜ, ሕዋጐ

Georgian (Mkhedruli) მხედრული `georgian` W◦ *Aa*◦

ოჟ შღრეფმშლწ ჟჟ, ზღო ეგ ჯეეუეჟზ ბთდ ჰოჟჯბ შიხნამდ, სჭნ ცუ ფრღ რყ

Greek Ελληνικό αλφάβητο `greek` W◦ *Aa*◦

Εψ λξτζθζαιγκωου ντζζ, ιζτζ Ωδ αοθτηηγ ζπν ιρβπμ γγηητγκξγ, πμλ τσλ γθυι σδ οψθαιωγπ

Diacritics are not included because the random output looks better without them.

Linear B `greek-linear-b` W◦ *Aa*◦

[illegible]

Hebrew אותיות דפוס `hebrew` W◦ *Aa*◦

רהגדוא סחיו קד תטש חרח סע עתהבחללו רש וק לגדה עישה, בעג קאל רתחיל רסתבגשלו קט ורעבה

Hebrew (cursive) *כתב יד* `hebrew-cursive` W◦ *Aa*◦

ושת דברב קט דהל תשק יפ אאאוליסרד אי לוא יהדש להדאו, הקלה חלגה חלדקכ לתחפסחוקוס יק גרחיצ

Paleo-Hebrew `hebrew-paleo` W° *Aa*°

𐤏𐤟𐤈𐤃𐤟𐤋𐤆𐤌𐤟𐤇𐤄𐤇𐤟𐤌𐤓𐤟𐤌𐤕𐤇𐤊𐤀𐤀𐤅𐤆𐤟𐤊𐤒𐤟𐤁𐤉𐤟𐤓𐤄𐤔𐤅𐤟𐤎𐤐𐤊𐤒𐤍𐤟𐤂𐤇𐤄𐤟𐤊𐤍𐤔𐤟𐤃𐤅𐤃𐤋𐤒𐤟𐤏𐤅𐤇𐤎𐤃𐤍𐤍𐤓𐤟𐤋𐤊𐤟𐤌𐤕𐤎𐤎𐤓𐤅

International Phonetic Alphabet [aɪ̯ pʰiː eɪ̯] `ipa` W° *Aa*°

ɵɠ ɡ͡bɯβɲ̥l̥n̥os̘ʊ ɛɑ, r̥d̪ʏ ɮr̥ d̪ʙʎœäyɞ o̞ʌɣ xɜɵħɯ ʏkʼʏɟœɪɣ, p̪e̞ʐ ɬɨ æ|pʼ ɢʟ e̞ʉt̪ʼɑɛo̞ɻə ʏr̥ b̪ʌʒβ̥̊ɘ

Hiragana ひらがな `japanese-hiragana` W° *Aa*°

えあ ゆさくほちょぴょおぢゃむ あか, まにう ぢゃびょ れにほかゆとじょ えまじ

Kanji (Kyōiku) 教育漢字 `japanese-kanji` W° *Aa*°

潮必 映課必居老里整害曜 週公, 粉専応 朗財 述笑田夏机成属 往届系 各号寺重統 期

All 1,026 *kyouiku kanzi* taught in elementary school.

Katakana カタカナ `japanese-katakana` W° *Aa*°

リツ モマザノレゴヘセチ ピュモ, ビュホギ ネシ コチウレカエコ シトヒ テストカ

Klingon (Latin script) `klingon-latin` W°

Ij vvbojIrpD ue, DQch ro ttlhepmrq vghS HQmvI SebuH’H, ’QD chl auo Qy SuSIDchav He oı

Klingon (pIqaD) `klingon-pIqaD` W° *Aa*°

Korean Hangeul 한글 `korean` W° *Aa*°

탓밋 뤼막멱곶왱떨밈둑언 덕걷, 뗩떤촬 람랠 뇨튕탯여첸맬쿰 나휨빰 깂바쇽툭뵤

This alphabet includes only blocks from the KS X 1001° character set, as the full Unicode set includes many blocks which are *theoretically* possible in Hangeul but are not found in Korean.

Latin (Classical) ABECEDARIVM LATINVM `latin` W° *Aa*°

FF GTMBOFCQVI GS, GRT ES EOAMBVV BYP VIVLX IYBSVYY, EIY VA YAD AO

Latin (bicameral w/ ⟨j⟩ and ⟨u⟩) Abecedarium Latinum `latin-bicameral` W° Aa°

Zt iafnalkzl fg, ena Sp xnuvnim hmn icxhu pybxzoy, aev yi uez ie ipndfege pi ahjlybo bp i

This is the default alphabet used if no ⟨alphabet⟩ argument is provided, in the tradition of *lorem ipsum*.

Mahjong tiles `mahjong` W° Aa°

🀰🀡🀫🀪🀦🀧🀩🀬🀭🀮🀯🀱🀲🀳🀴🀵🀶🀷🀸🀹🀺🀻🀼🀽🀾🀿🁀🁁🁂🁃🁄🁅🁆🁇🁈🁉🁊🁋🁌🁍🁎🁏🁐🁑🁒🁓🁔🁕🁖🁗🁘🁙🁚🁛🁜🁝🁞🁟🁠🁡🁢🁣🁤🁥🁦🁧🁨🁩🁪🁫🁬🁭🁮🁯🁰🁱🁲🁳🁴🁵🁶🁷🁸🁹🁺🁻🁼🁽🁾🁿🂀🂁🂂🂃🂄🂅🂆🂇🂈🂉🂊🂋🂌🂍🂎🂏🂐🂑🂒🂓🂔🂕🂖🂗🂘🂙🂚🂛🂜🂝🂞🂟🂠🂡🂢🂣🂤🂥🂦🂧🂨🂩🂪🂫🂬🂭🂮🂯🂰🂱🂲🂳🂴🂵🂶🂷🂸🂹🂺🂻🂼🂽🂾🂿🃀🃁🃂🃃🃄🃅🃆🃇🃈🃉🃊🃋🃌🃍🃎🃏🃐🃑🃒🃓

Mathematical symbols `math` W° Aa°

{∋ ⁷¹828⁵˙·.∮d ∨m,)27 △j ↦^7=3}≅ ÷-∴ 9≠∴=⎣ a∉4ℵΔΣ8, ∂\⋊ p: ∬^} Ød 7+⌋s↦8}ℍ 5= ℍ⊐

Mongolian ᠮᠣᠩᠭᠣᠯ ᠪᠢᠴᠢᠭ `mongolian` W° Aa°

᠕ ᠮᠣᠩᠭᠣᠯᠤᠰᠤᠨ ᠤᠭ, ᠬᠡᠯᠡᠨ ᠳᠤ ᠬᠡᠪᠯᠡᠯᠴᠡᠭᠡᠨ ᠦᠭᠡᠶ ᠵᠣᠬᠢᠶ ᠰᠢᠨᠵᠢᠯᠡᠭᠡᠳᠦ, ᠰᠢ ᠬᠣᠭ ᠤᠷᠤᠨ ᠳᠤ ᠪᠢᠴᠢᠭᠳᠡᠨᠡ

Currently vertical text is yet to be supported in Typst°. Mongolian is written in columns going from top to bottom and ordered from left to right (tb-lr). Rotating a text 90° clockwise can work, but lines have to be separated into individual rotated boxes:

ᠮᠣᠩᠭᠣᠯ ᠪᠢᠴᠢᠭ
ᠬᠡᠪᠯᠡᠯ ᠪᠢᠴᠢᠭ
ᠣᠷᠣᠨ ᠬᠡᠪᠯᠡᠯ

See `ogham` for another vertical script.

Morse code `-- --- ·-· ··· · -·-· --- -·· ·` `morse` W° Aa°

·- ····-···· -·- -·- -- ·---·-- ·-- -·-·· ----·-·· --- ·- ··- -· ·--·--·· --

Maze pattern (1) `maze-1` W° Aa°

\\ \/\\\/ \\, /// /\ \/\\/\ /\\ /\\\ \/\//, \// // \// \ /\/\/

The following alphabets yield nice patterns when used in a multi-line setting, like so:

Note that the ⟨leading⟩ of `#par()` might require some tweaking so the lines meet.

This pattern can be generated in Commodore 64 BASIC by a one-liner — `10 PRINT CHR$(205.5+RND(1)); : GOTO 10` — which is indeed rather poetic°.

Maze pattern (2) `maze-2` W° Aa°

\\ \\/\/\\/ //, ///\ //\ \/\\\\ \/\ \\/\\ //\\\\, ///\ \\ /\/ /\ /\\\/\/

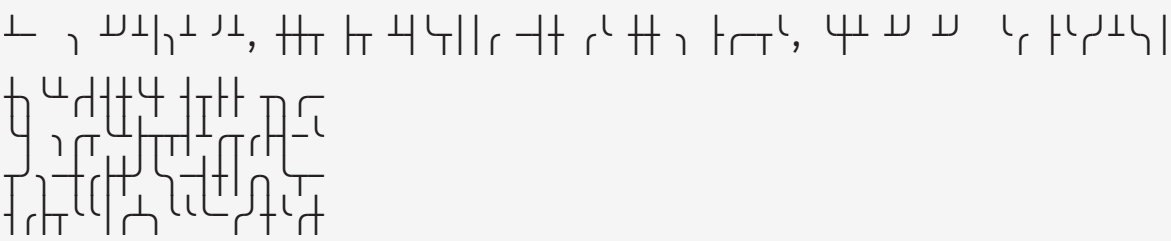
Maze pattern (3) `maze-3` W° Aa°

Maze pattern (4) `maze-4` W° Aa°

\/ ((^\/\/\ \/), \/(\) \/\//\/^\ ( ) ((/\ \/\/\/\, ^\/ (\ (\^) \/ /\\/\(

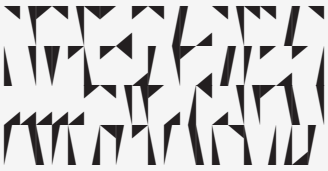
This one looks like actual writing, slightly reminiscent of Baybayin (`baybayin`).

Maze pattern (5) maze-5 W° Aa°

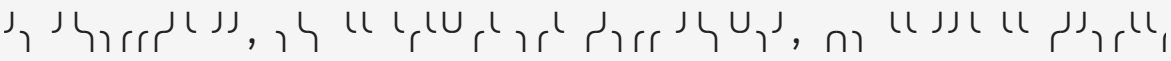


Maze pattern (6) maze-6 W° Aa°

The pattern below is reminiscent of *pIqaD* (klingon-pIqaD).

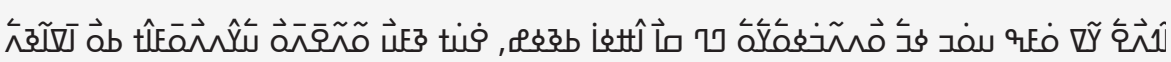


Maze pattern (7) maze-7 W° Aa°



The pattern below is reminiscent of *Mkhedruli* (georgian).

N’Ko ߒߞߏ nko W° Aa°



Binary digits numerals-bin W° Aa°

```
01 110110000 11, 100 11 1110110 110 11011 0101011, 000 01 011 11 10001000
```

Decimal digits numerals-dec W° Aa°

```
37 232482558 24, 866 81 4583758 375 32916 8483177, 228 55 413 81 69218904
```

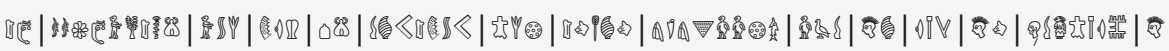
Hexadecimal digits `numerals-hex` W° Aa°

61 640A91643 69, 670 27 7F13C7C DF9 B2A9A 4C29AF1, 6C4 90 C4C F1 1C9DF6A1

Ogham ᚛ᚑᚏᚐᚋ᚜ `ogham` W° Aa°

Forfeda are not included. Ogham is quite unique in that it goes from the bottom of the writing material upwards; although is yet to be supported in Typst°, one can rotate a text 90° anticlockwise. This doesn't work well with long texts, but historically Ogham inscriptions are quite short anyway. See `mongolian` for another vertical script (tb-lr).

Phaistos Disc `phaistos` W° Aa°



Polish alfabet polski `polish` W° Aa°

Iu sńuebgdzis ęó, cłó Oy uącidżyg cie oóipę żcrzwzhdz, nńb śk gtą dj hyąehóóc ął nadchnj

Redaction `redaction` W° Aa°

Shavian (English) 𐑣𐑦𐑮𐑤 𐑡𐑡𐑮𐑤𐑣𐑤 `shavian` W° Aa°

𐑝𐑟 𐑡𐑪𐑤𐑹𐑕𐑮𐑣𐑪 𐑤𐑪, 𐑡𐑨𐑯 𐑮𐑟 𐑓𐑓𐑡𐑨𐑑𐑒𐑡 \𐑪𐑝 𐑖𐑩𐑮𐑪 <𐑟𐑨𐑷𐑩𐑝, 𐑟𐑛𐑤 𐑖𐑤 𐑒𐑻 𐑟 𐑒𐑻 𐑮𐑪𐑜//𐑯𐑻 𐑪𐑮 𐑮\𐑻𐑜𐑮𐑪 𐑤\

sitelen Lasin sitelen Lasin `sitelen-lasin` W°

wan seli akesi pimeja tonsi tu kon wan olin monsi ante pona ko selo a li kokosila olin mar

This one is a bit different, since it doesn't contain letters but whole words in the Latin script (the whole toki pona dictionary…); this makes the output much longer.

sitelen pona 🜲 `sitelen-pona` W° Aa°

sitelen sitelen 𖹠𖹠 `sitelen-sitelen` W° *Aa*°

Tengwar (Beleriand) ᵽıɑ ᵽɯϽɤɣıcᵽ̄ `tengwar-beleriand` W° *Aa*°

ɦϽ ᵽbɦɖöɗɖq̄ɲɑl ɪɑɒ ć𝼄ᵽ̄ ıc̈ ẍoɑc̈óɑ́ẍ ɗᵽ̄ᵽ̄ ɤnᵽ̄ɑ́ɤɯ c̈onᵽ̄ɦhᵽᵽcɤ ɦɯɤɯ ɣᵽ ᵽ̄ᵽ̄ɯɥ ɤɯ̄ ɣ

This broadly aligns with the mode of Beleriand°, with compromises.

Tifinagh ⵜΣⵣΣⵏⵑ `tifinagh` W° *Aa*°

ⵔ⵿ ⁖⁖--<ⵅⵢⵗⴼᵘⵅ ⵗⵟ, ΔØ⁝ ⵣ⁝ ⵅⵏⵔⵜØⵅⵣ --ⵗⵣ ⅄ⵑⵎ<ⵅ ⵗⵔⵅⵔ⵿--ⵄ, ⵔⴹⵗ ⵗⵔ ⵣⵜⵜⴼ ⁘ⵗ ⁖ₒⵅ⁘+ⵊ< ⅄ⴹ

Empty space emulating correction fluid `tippex` W° *Aa*°

, ,

Ugaritic `ugaritic` W° *Aa*°

Ukrainian Українська абетка `ukrainian` W° *Aa*°

Що нцвбдосіу ад, ини Їі кєєїтои їєі асютм ьояяиец, аіь хи ґмщ ец лїєґаафг уу чбнхєєф

Underscore `underscore` W°

_ _____ _, _ _ _____ _ ____ _____, _ _ _ _ _____ _ _____ _ ____. _ _ _

Vietnamese chữ Quốc ngữ `vietnamese` W° *Aa*°

Ềb ghũỷmúộịợs chk, nhkhễ Rỡ ứcnquữãqu ếbch nghầóểtr gisgcõựầ, ộcc êử giýth uì ngaõó

Voynich `voynich` W° *Aa*°

Welsh Yr wyddor Gymraeg `welsh` W° *Aa*°

Êo ddeŵoeegphchn aei, wywŷ Cr ôouawâowyuw yoeoi oeaeuoioi aiidaeuâae, mgi ôoe eufc

Part III.
Index

B

#baffle **3**, **5**, **8**, **9**, **10**, **11**

O

#outline **11**

P

#par **18**