Manual for the quetta (𝔔𝔲𝔢𝔱𝔱𝔞) module

Florent Michel

version 0.1.0

Contents

1. Introduction

1.1. 'Quetta'?

'Quetta' (𝔔𝔲𝔢𝔱𝔱𝔞) means 'word' in Quenya [1][1], one of the fictional languages invented by British writer and philologist J. R. R. Tolkien. It thus seemed fitting for a module aimed at making the process of typing these languages easier.

Words are also, loosely speaking, the base units this module works on, as we shall see in more details below. While its general philosophy is to map each symbol used in Tolkien's elvish languages to letters from the Latin alphabet, a few word-wise substitution rules were implemented so that, in *most* (but probably not all) cases the correct spelling can be obtained by typing the word phonetically. For the same reason, the mapping generally works on groups of letters rather than individual ones, at least when there is no natural one-to-one mapping between individual symbols.

1.2. The Tengwar script

A proper introduction to Tengwar is way beyond the scope of this document. We refer interested readers to Appendix E of the book [2] and online references such as Wikipedia, tolkiengateway.net, omniglot.com, or tecendil.com.

In short, Tengwar (𝔱𝔢𝔫𝔤𝔴𝔞𝔯 in Quenya mode) is one of the scripts invented by Tolkien, primarily consisting of 36 letters (called *tengwar*; singular: *tengwa* (𝔱𝔢𝔫𝔤𝔴𝔞)) and diacritics (*tehtar* (𝔱𝔢𝔥𝔱𝔞𝔯 ; singular: *tehta* (𝔱𝔢𝔥𝔱𝔞))). There are several ways to relate tengwar to sounds, called *modes*. This module primarily focuses on the Quenya (𝔔𝔲𝔢𝔫𝔶𝔞), or 'classical', mode, in universe the original way to write tengwar. Support for the other modes described by Tolkien is planned for a future version.

2. How to use

2.1. Requirements

- Typst version 1.11.0 or up
- The Tengwar Annatar fonts version 1.20 (support for other Tengwar fonts is not currently planned).

[1] https://www.elfdict.com/w/quetta

2.2. Importing the module

To import the module, simply add

```
#import "<path>/quetta.typ": *
```

at the top of your `.typ` file, where `<path>` is the path to the quetta module.

2.3. Design principles

This module provides one main command for each supported mode—at the moment, only `quenya` is implemented. This command takes text (possibly including formatting) as input and performs the following sequence of operations (not necessarily in this order):

1. Phonetic translation into tengwar and tehtar—for instance, converting `quenya` to .
2. Application of spelling rules—for instance, converting to .
3. Conversion of numbers in base 12 and conversion to the tengwar number system (see below)—for instance, `144` becomes .
4. Conversion of punctuation symbols—for instance, `?` becomes .
5. Adjustments to the position of tehtar and to the kerning between some symbols.

Alternative glyphs, when available, can be obtained with the symbol `£`:

`quenya[s]` `quenya[£s]` `quenya[ss]` `quenya[£ss]`

`quenya[sa]` `quenya[£sa]` `quenya[ssa]` `quenya[£ssa]`

For tengwar associated with a sound starting with ‘k’, the standard glyphs are obtained using the spelling ‘c’ for *calma* () or ‘qu’ for *quessë* (), and the alternatives glyphs with a ‘k’ or ‘kw’:

`quenya[c]` `quenya[k]` `quenya[qu]` `quenya[kw]`

Formatted text is supported, although it is still somewhat experimental:

`quenya[quetta *quetta* _quetta_ _*quetta*_]`

For a larger amount of text or more involved formatting, it can be easier to use a `show` rule as follows:

```
#[#show: quenya
  quenya
  #h(1em) *quenya*
  #h(2em) _quenya_
]
```

One limitation of the current implementation is that functions changing other style properties such as text color must be called *after* the conversion function. For instance, a centred 16-points italic version of the Quenya word ‘tengwar’ with a blue-green linear gradient may be obtained as follows:

```
#set align(center)
#text(size: 16pt,
      fill: gradient.linear(blue, green)
     )[#box(quenya[_tengwar_])]
]
```

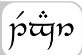
2.4. Quenya

2.4.1. Generalities

The implementation of the Quenya mode mostly follows Reference [3], summarizing information available in Appendix E of [2] and examples provided in other parts of the book. In universe, the Quenya language and pronunciation evolved somewhat between its origins and events described in [2]; in this implementation, we aim to stay close to the Second Age conventions. Here are a few basic examples:

`quenya[quenya]` 

`quenya[quetta]` 

`quenya[tengwar]` 

`quenya[namárië]` 

A full description of the Quenya mode is beyond the scope of this document. As a first approximation, consonant sounds are represented by *tengwar* as follows[2]:

consonant	tengwa
t	
nd	
th	
nt	
n	
r	

consonant	tengwa
p	
mb	
f	
mp	
m	
v	

consonant	tengwa
c	
ng	
h	
nc	
n	
y	

consonant	tengwa
qu	
ngw	
hw	
nqu	
ñw	
w	

consonant	tengwa
ñ	
rd	
l	
ld	
s	
ss	

[2]In these tables, the letter 'u' represents the consonant sound 'w'.

Different tengwar are used for the same sounds in different situations; see Section 2.4.2. Vowel sounds are generally represented by a *tehta*, placed either on the previous consonant or a short carrier for a short vowel, or a long carrier for a long vowel[3]:

vowel	short version	long version
a		
e		
i		
o		
u		

Diphthongs of the form -*i* and -*u* are obtained by adding a theta to an ‘i-glide’ or ‘u-glide’ symbol:

ai	oi	ui	au	eu	iu

2.4.2. Substitution rules

The equivalences mentioned in Section 2.4.1 should give an approximate phonetic transcription from the Latin alphabet to tengwar in the Quenya mode. This does not mean, however, that the spelling is correct. Further substitution rules are required for that; see for instance Reference [3].

- As briefly mentioned above, a short vowel sound following a consonant sound is written as a tehta on the latter. For instance, *númen* (*west*) is written , with replacing . This does not apply to long vowels, the second consecutive vowel after a consonant, or (obviously) to a vowel sound at the start of a word.
- If is followed by a vowel or *y* and a vowel, it is replaced by .
- If follows a consonant, it is replaced by two dots under the corresponding tengwa: → .
- If a single follows a consonant, it is replaced by an ‘s-hook’: for instance, → .
- If is followed by a short vowel, it is reversed: → .
- If is followed by a short vowel, it is reversed: → .
- If is part of the consonant cluster *hl* or *hr*, it is written with a vertical bar as in *hrívë* (*winter*): .
- If is not part of the consonant cluster *hl* or *hr*, nor at the start of a word, it is replaced by .
- Two successive identical tengwar with no tehta on the first one are replaced by a single tengwa with a wavy line under it. For instance, the word *anna* (*gift*) is written , with replacing two s. Similarly, in (*quetta*), replaces two s.

2.4.3. Capital letters

There is, as far as I am aware, no standard way to write capital letters in Tengwar. One option is to use bold to denote a capital letter:

```
quenya[#strong[Va]limar]
```

2.4.4. Punctuation

The module provides the following punctuation symbols:

[3]We use an acute accent to denote long vowels. For instance, a is rendered as and à as .

input	,	.	-	—	!	?	(	)	/
output	·	:	~	~	[illegible]	[illegible]	(	)	[illegible]

Note: Generally, parentheses in Quenya are denoted by the single symbol [illegible]—there is no distinction between opening and closing parentheses. We deviate from this convention by mapping the symbol '(' to (and ')' to). The proper Tengwar parenthesis is mapped to '/'.

End-of-paragraph symbols can be obtained by combining commas and periods:

`quenya[.-]` :~ `quenya[.,]` :· `quenya[..]` :: `quenya[,.,]` ·:·

The decorations [illegible] and [illegible] are obtained using the French quotation marks '»' and '«':

`quenya[»quenya«]` [illegible]

The symbol ':' can be used to prevent glyph combination:

`quenya[nn n:n]` [illegible] `quenya[na n:a]` [illegible]

2.4.5. Number system

Quenya uses a base-12 system, with 12 digits listed in the following table:

0	1	2	3	4	5	6	7	8	9	10	11
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

In contrast with the usual system, multi-digit numbers are written (from left to right) from least to most significant digit.

Examples:

`quenya[1]` [illegible] `quenya[11]` [illegible] `quenya[12]` [illegible] `quenya[100]` [illegible] `quenya[123]` [illegible]
`quenya[144]` [illegible] `quenya[1000]` [illegible] `quenya[10000]` [illegible] `quenya[65537]` [illegible]

2.4.6. Example: Namárië

One of the most famous texts in Quenya is the poem *Namárië* (), originally written in Reference [2][4] and available for instance in Reference [4]. Below we show the same text without (left) and with (right) the `#show: quenya` command. We use a spacing between line of 0.7em to clearly separate them (some tengwar have a relatively large vertical extension).

Namárië

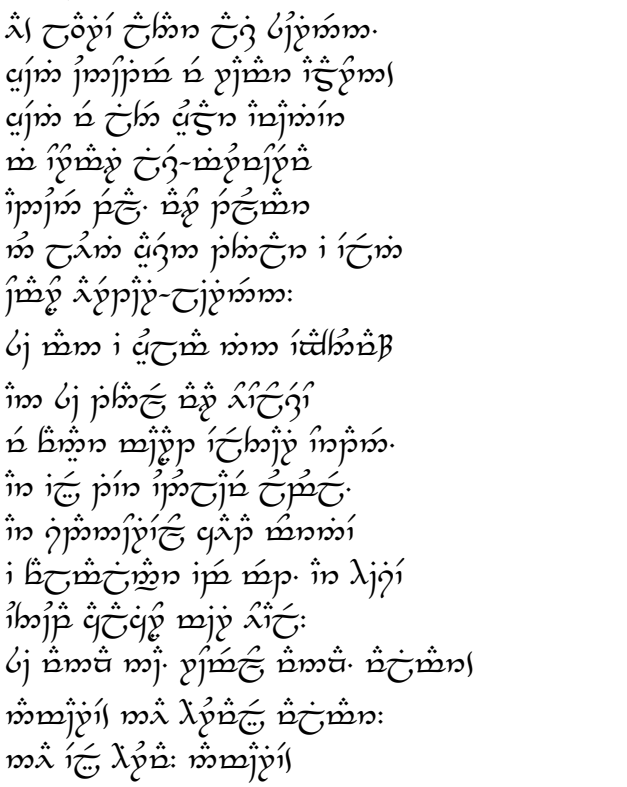
Ai! laurië lantar lassi súrinen,
yéni únótimë ve rámar aldaron!
Yéni ve lintë yuldar avánier
mi oromardi lisse-miruvóreva
Andúnë pella, Vardo tellumar
nu luini yassen tintilar i eleni
ómaryo airetári-lírinen.

Sí man i yulma nin enquantuva?

An sí Tintallë Varda Oiolossëo
ve fanyar máryat Elentári ortanë,
ar ilyë tier undulávë lumbulë,
ar sindanóriello caita mornië
i falmalinnar imbë met, ar hísië
untúpa Calaciryo míri oialë.
Sí vanwa ná, Rómello vanwa, Valimar!

Namárië! Nai hiruvalyë Valimar.
Nai elyë hiruva. Namárië!

****



[4]Book 2, ch. 8 “Farewell to Lórien”

2.5. The One Ring inscription

Although the Black Speech is not implemented yet, the One Ring inscription can be reproduced using the Quenya mode as follows:[5]

```
quenya[
  _»Ka:nssangw:nd£rombta£lo£kwô, Ka:nssangw:ngwmbetalo«
  #linebreak()#v(0.7em)
  Ka:nssangwth£rquata£lo£kwô, £Ngwa:mb£rossmokii:qu£rpetalo_
]
```

Obviously, that's not quite how the ring inscription is supposed to sound. One reason is simply that the Quenya and Black Speech modes have different relations between symbols and sounds: to obtain the same written result, one has to 'transcribe' the phonetic description to how it would be read in the Quenya mode. Another difference is that some of the tengwa forms used in the ring inscription are generally not used in Quenya; we thus use the symbol `£` to get variants. We also use `£` to switch between [illegible] and [illegible]. Finally, words are separated with `:` to avoid repeated consonants being combined. Here is the result, with a colour gradient in the background to mimic a golden surface and on the text to represent incandescence:

The full code for the above example is:

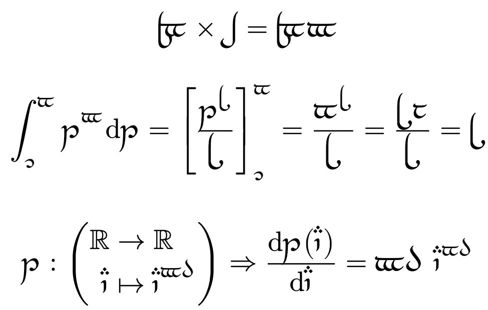
```
#let ring-inscr-quenya = "
  _»Ka:nssangw:nd£rombta£lo£kwô, Ka:nssangw:ngwmbetalo«
  #linebreak()#v(0.7em)
  Ka:nssangwth£rquata£lo£kwô, £Ngwa:mb£rossmokii:qu£rpetalo_
"
#set text(top-edge: "ascender", bottom-edge: "descender",
          fill: gradient.linear(rgb(150,0,0), rgb(100,20,0), rgb(255,0,0), space: rgb, angle:
20deg))
#align(center, block(
   fill: gradient.linear(rgb(157,103,7), rgb(250,250,152), rgb(157,103,7), space: rgb, angle:
80deg),
  inset: (top: 1em, left: 1em, right: 1em, bottom: 1.5em),
  radius: 5pt,
  quenya(eval(ring-inscr-quenya, mode: "markup"))))
```

[5] This is obviously a bit of a hack, meant only to show how the limitations of having only one mode implemented can be circumvented. This example is not supposed to be stable and might render differently in a later version.

3. Math mode?

Use of tengwar in math mode is not fully supported, although it should work in simple cases. In math mode, you'll need to apply the conversion function to each part of a formula you want to write in Tengwar, which can be made slightly less cumbersome by redefining it to a shorter command:

```
#let q = quenya
$
  #q[6] times #q[7] = #q[42]
$
#v(1em)
$
  integral_#q[0]^#q[2] #q[t]^#q[3] upright(d)#q[t]
  = [ #q[t]^#q[4] / #q[4] ]_#q[0]^#q[2]
  = #q[2]^#q[4] / #q[4]
  = #q[16] / #q[4]
  = #q[4]
$
#v(1em)
$
  #q[t] :
    mat(delim: "(",
      RR & -> RR ;
      #q[a] & |-> #q[a]^#q[123])
  =>
  (upright(d)#q[t]lr((#q[a]))) / (upright(d)#q[a]) = #q[123 a]^#q[122]
$
```

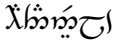


Writing math-heavy content in tengwar would probably require a specific module, though, as well as a different tengwar font designed for this purpose (or a math font designed to work well with a tengwar font).

4. How to contribute

Any kind of contribution is warmly welcome! Here are a few ways you can help:

- **Bug reports:** Some text rendering incorrectly in Tengwar? Unexpected formatting? Any other issue with the code or documentation? Please report it! This module was only tested on a very small corpus so far, and identifying any corner case where it does not work as intended is very useful!
- **References:** There is a lot of content available, both online and in printed books and magazines, about the languages invented by Tolkien, how they relate to his works, and their relevance in today's cultural fabric. I am unfortunately not very familiar with them; but if you know good references please let me know and I'll cite them.
- **Language help:** My knowledge of Tengwar and the languages invented by J. R. R. Tolkien is quite superficial, and I may well have missed or misunderstood some of the rules for writing in Tengwar. If you spot anything that looks wrong, please let me know!
- **Implementation:** The Typst code is likely not quite as efficient nor as clean as it could be. If you can see better ways to implement something, please feel free to let me now or to submit a pull request with an improved version!
- **Feature requests:** Any feature request is welcome. I can't promise I'll have the time and knowledge to implement everything that would be nice to have; but if you'd like to see something implemented please let me know—or submit a pull request if you've already implemented it!



Bibliography

[1] "elfdict.com." [Online]. Available: https://www.elfdict.com/
[2] J.R.R. Tolkien, *The Lord of the Rings*.
[3] "Quenya Tengwar – A mini course by Eruantalincë." [Online]. Available: https://www.councilofelrond.com/languages/QTL.htm
[4] "Namárië." [Online]. Available: https://tolkiengateway.net/wiki/Nam%C3%A1ri%C3%AB