WIP Paper Outline

Steven Tammen

June 16, 2018

Contents

1 Section 1: Why?

1.1 Why this project?

1.1.1 The lack of open source, *customizable* software

1.1.2 The lack of software that works for nonstandard keyboard layouts

1.1.3 The lack of software that bundles multiple language layouts together

1.2 Why this paper?

1.2.1 Justifying design choices

1.2.2 Creating a starting point for people that may have different opinions than myself

2 Section 2: Nuts and bolts

2.1 Keyboard layouts

2.1.1 Letters

2.1.2 Context-specific/alternate letter forms

2.1.3 Mandatory markup: vowel points, diacritics, etc.

2.1.4 Additional markup: metrical marks, cantillation marks, etc.

2.1.5 Punctuation; language-specific symbols

2.2 Unicode

2.2.1 History, scope, and purpose; peculiarities

2.2.2 Precomposed and decomposed Unicode

2.2.3 Combining multiple diacritics

2.3 Fonts

{Todo: [1]}

2.3.1 An overview of existing options (for Greek and Hebrew)

- SBL Greek and Hebrew

[1] Ideal font design + discussion

- Gentium Plus and Ezra SIL
- Cardo
- New Athena Unicode
- Google Noto Font. Research.

3 Section 3: The Unicode Language Layers project

3.1 Sane defaults combined with ease of use

3.2 Customizability as a first order priority

- Thorough API
- In-line comments
- Examples in the form of Greek and Hebrew layers

3.3 Minimal interference with normal computer use

- Quick and easy on and off
- Consistent keyboard shortcuts (languages do not interfere with normal shortcuts)
- Leader-prefixed punctuation for normal behavior (for when punctuation gets hijacked by a layer for diacritics and so forth)

3.4 Consistency across multiple languages

3.4.1 For end users

- Base markup for Latin, German, French, Italian, Spanish. Leader-prefixed diacritics.
- Switching between different alphabets; using different alphabets

3.4.2 For designers

- Consistent handling of precomposed and decomposed Unicode
- Abstracted, language-blind functions to extend to new languages with minimal effort
- If you understand how to code a layer for one language, you should be able to code layers for other different languages.

4 Section 4: Greek as an example

4.1 Letters

4.1.1 The relationship between memorability and speed

4.1.2 Native-language layouts in muscle memory

4.1.3 Issues in constructing associations

4.1.4 A Greek-English keymap

4.2 Context-specific/alternate letter forms

4.2.1 Final sigma

4.2.2 Lunate sigma

4.3 Mandatory markup

4.3.1 Breathings

- smooth, rough
- vowels and rho

4.3.2 Accents

- acute, grave, circumflex

4.3.3 Iota subscripts

4.3.4 Diaeresis

4.3.5 The koronis

4.4 Additional markup

4.4.1 Vowel quantity: macrons and breves

4.4.2 The underdot

4.5 Punctuation; language-specific symbols

{Todo: [2]}

4.5.1 Question marks and semicolons

4.5.2 A discussion of "hybrid" punctuation, and accessing normal punctuation when desired

5 Section 5: Hebrew as an example

5.1 Letters

5.1.1 Handling cases of identical letter sounds

5.1.2 A Hebrew-English keymap

5.2 Context-specific/alternate letter forms

5.2.1 Word final letters: the sofit forms

5.2.2 The Begadkephat letters

5.2.3 Shin and Sin

5.3 Mandatory markup

5.3.1 A note about opinionated design decisions

- "Case study" – the *matres lectionis* letters. Automatically including vav and yod when they are vowel indicators.

[2] Metrical marks, special numerals, drachma symbol

5.3.2 Basic vowel points

5.3.3 Shva and reduced vowels

5.3.4 The dagesh

5.4 Additional markup

5.4.1 The meteg

5.4.2 Cantillation marks

5.5 Punctuation; language-specific symbols

5.5.1 A discussion of languages that use "mostly normal" punctuation (from the English point of view)

5.5.2 The geresh

5.5.3 The gershayim (lit. "double geresh" – this word is plural)

5.5.4 Colon and *sof pasuq*

5.5.5 Vertical bar and *paseq*

5.5.6 Hyphen and *maqaf*

5.5.7 Shekel symbol

6 Section 6: Efficient typing practice for non-native languages

6.1 Introduction to efficient typing

6.1.1 Practicing based on word frequency

6.1.2 Practicing based on N-gram frequency; affixes

- (Derivational) Morphemes rather than words as a training focus

6.1.3 Abbreviating very frequent words and phrases

6.1.4 Practicing the sorts of texts you are going to type

6.2 Creating necessary resources

6.2.1 Word frequency tables

- Perseus, TLG, handling overlapping forms

6.2.2 N-gram frequency tables

- Similar process. Handling semantic boundaries in regexes? How to automate morphological analysis without obvious delimiters like spaces for words?

6.2.3 Area-specific practice texts

- Downloading from free/uncopyrighted sources. Perseus, Project Gutenberg.[3]

6.3 Typing practice

6.3.1 Amphetype

6.3.2 Lesson generation from frequency tables and practice texts

6.4 Crossover benefits

6.4.1 Vocabulary lists by frequency for specific domains

6.4.2 Morphological analysis and generative vocabulary

- Prefixes, suffixes, and roots. Developing an eye for picking up meanings automatically, simply by knowing what different parts of the word mean in general.

[3] Automate with script? Probably also outside scope of project.

7 Section 7: Pedagogical applications

7.1 Orthography for digital natives

7.1.1 Standardization of letterforms

- Reducing the learning load in the first few weeks of Hebrew: block scripts and cursive scripts.
- Possible in handwritten as well (just only writing in block)

7.1.2 Typing speed and writing speed

7.1.3 But the permanence of handwriting

- Tests

7.2 Examples of typing-related pedagogical aids for Greek

7.2.1 Learning the accentuation system

- Practicing the typing of accents while learning about the rule of contonation, morae, and recessive accents.

7.2.2 Common irregular verbs

- Practicing the typing of certain very common irregular verbs (like *eimi*, e.g.) while simultaneously learning their paradigms.

7.2.3 Practicing reading/speaking Greek; ”reading by typing”

- Practicing typing in general by pulling in Greek texts from Perseus as typing training material. Students could be encouraged to also read the texts out loud as they type them. (Not necessarily understanding the Greek, but getting to see how it sounds and flows).

8 Section 8: Concluding remarks

8.1 Specific implementation benefits

8.1.1 Who should make the switch to this system? Is this project really worthwhile?

8.1.2 The low opportunity cost for the next generation

8.2 Moving forward with more languages

8.2.1 Current project: focus on Greek with Hebrew as a foil

8.2.2 Possibility to expand much further

8.3 Suggestions for further research

8.3.1 Corpus generation

8.3.2 Morphological analysis

8.3.3 Graphical frontends for customization

8.3.4 System APIs for keystream manipulations *across platforms*

8.3.5 AI autograders for language exercises

9 Section 9: Appendix

9.1 Integrating general electronic/online resources into classes

9.1.1 Language input as a pain point

- A lack of good keyboard input is a significant damper to the use of electronic/online resources.

9.1.2 The value of electronic/online resources

1. Elecronic lexica and morphology parsers

   Dangers of over-reliance, but great benefits all the same. Arbitrary searches (those that require the ability to type native text) can be necessary when using paper sources rather than cross-linked sources like those on Perseus.

2. Searches

- Fuzzy search (i.e., lemma search), finding passages and references, searching on word usage or specific form.
- Searching typed notes, if people type class notes

3. Electronic flashcards

   More polarizing whether or not they are useful, but making them easier to construct is definitely a good thing. Spaced repetition studying, Anki.

4. Autograded sentences

   - Practicing typing in general by providing form-fields to enter sentence translations. Depending on the difficulty of implementation, it might be possible to create an autograder for practice sentences in Athenaze, for example. If care was taken to follow vocabulary acquisition (so as to limit the lexicon input for the program and make it deterministic), it would be easy for professors to design supplemental/optional practice exercises that the students could complete with instant feedback and no extra work for the professor.

9.2 Word Processing

9.2.1 Reasons why something other than Word might be desirable

- Automatic font use rather than manual switching

9.2.2 Example: Emacs' Org mode to PDF using XeLaTeX

- Support for RTL languages and automatic display
- Polyglossia
- Automatic font switches

9.2.3 Yudit?

{Todo: [4]}

[4] Need to research more.

9.3 Abbreviations

- More of a personal thing. Can algorithmically generate in theory. (Outside scope of this project).
- Probably good to look at the 10 or 15 most common words and see if anything jumps out at you
- Creating regex hotstrings in this particular AHK implementation