Contents

1 Summary of progress that has been made

The two main things that have been accomplished in this first leg of the project are as follows:

1. Add an initial section to the paper describing layout design. Create tentative Greek keymap.
2. Code up a working version of the Greek layer, with full diacritic support accomplished through punctuation correspondences.

1.1 Paper section

Ironically, I had significant problems getting my LaTeX PDF exporting working with Greek Unicode. No matter what packages I put in the header or which compiling backend I used (e.g., pdflatex, xelatex, etc.), I could not get Greek characters to properly display. I spent a few hours trying to debug this, then posted a question on a Stack Exchange and gave up. Hopefully I'll get this resolved in due time, but for the moment, people who wish to read the paper best stick with GitHub's rendering of the Org file.

I tried to not get too technical with regard to keyboard layout optimization, but still give people an idea of what is involved. The main takeaway is that it makes the most sense to piggyback off of already-learned layouts rather than attempt to get an entirely foreign layout in muscle memory, even if the latter might be "better" from a purely theoretical perspective (i.e., have lower finger travel distance, lower same finger, etc.).

The first stab at a Greek-English correspondence is now present. I used phonetic correspondences as much as possible, and then other mnemonics after that. Only the letter Theta proved impossible to place with such things. A brief discussion of Koppa and Digamma (to go on Q and V, respectively), is also included.

There are some TODOs present, and I need to figure out how to set up footnote exporting within Org mode. But this is a reasonably complete start for the Greek side of the design process.

1.2 Code

Because of the AutoHotkey software package I am using (Dual), the basic remappings proved to be trivial. It was the diacritics that proved challenging (as expected).

The thing that took me the longest this first go of it was figuring out a seamless way to add and remove diacritics. I opted to go with precombined Unicode characters at the moment, but have read enough now to see the pros and cons of both the precombined and decomposed approaches. This page in particular was very informative about "why things are the way they are," and exactly how Greek works with Unicode. Even though decomposed support is getting better (and works absolutely fine with some fonts and applications), it looks like precombined is still the most universally "safe" option for input.

At a high level, my current implementation uses a few global status variables to keep track of which diacritical features are present (e.g., breathings, accents), and then a whole bunch of conditionals to output the correct precombined characters. The simplest way to get this working was simply to backspace the vowel (with XY diacritics, e.g.), and replace it with a new version of the vowel (with XYZ diacritics, e.g.). From the end user's perspective, it looks as if things are getting added and removed instantly, even though the old character is getting deleted, and the new character displayed.

Having all the logic in branched conditionals keeps the checks performant, and I haven't observed any noticeable latency.

2 Summary of things relating to the progress that still need polishing

2.1 Paper section

Aside from getting the PDF exporting working, I need to add justification for the punctuation symbols I chose for diacritics ('‘~/[] for acute, grave, circumflex, iota subscript, rough breathing, and smooth breathing, respectively).

These were chosen because of a combination of memorability (a backtick looks like a grave accent, for example), and efficiency (on QWERTY, square brackets don't require a shift but parentheses do, for example).

2.2 Code

I still need to add rough breathing for Rho, and add an option to make backspacing incremental for diacritics (right now it deletes the whole vowel). You can get rid of diacritics you don't want by pressing the key that corresponds to them again, but some people might want Backspace to do the same, since it is somewhat more logical.

I also need to add support for dieresis (to show syllable boundaries in ambiguous vowel combinations), and macrons/breves for vowel quantity. The issue with these is that there are not "official" precombined Unicode characters for many combinations. Whether I choose to use decomposed diacritics or Private Use Area (PUA) code points (e.g., see here), implementation is going to be a real headache.

I am also debating whether or not to even try and create options for writing diacritics through sequences and key combinations, since these would be even more complex than the current method, and far less efficient, as far as I can see. Of course, I'm opinionated (some individuals in the survey did wish to use these options), but even so, I may opt to make a strong argument for why I have things set up how they are (to try and "convert" people) rather than spend a bunch of time implementing what amounts to an inefficient entry method. I could also write a brief section on how it *could* be done, and then leave implementation up to someone else, if they wished to have such functionality.

3 Other things moving forward

Before getting much further on the Greek end of things, I'll want to get much of the same done for Hebrew. That is, create a keymap and code the basic implementation. Of course, Hebrew is even more challenging on the implementation side of things since all the vowels are done through diacritics.

I also want to write up an API and fairly extensive documentation for how things may be changed. I'm toying with the idea of a GUI interface via JavaFX or QT (etc.) to let people configure things without having to look at any code, but honestly, this would take a pretty substantial amount of time to set up. I think it might be best for me to document everything (the functions and their use, primarily), give some examples, and then let people customize things in the scripts themselves, with support through a Google Group or other mailing list. We will see.