Future Tai Heritage Pro Modifications

It's quite possible that the version of Tai Heritage Pro that we are currently working on is the last version we will ever do. But in the event that we do need to revise it again, here are some ideas that I have for the next version. (JB)

1. OT glyph group cNarrowCons_FollowsO_Bold

UniAAAE and uniAAAF should not be included in this group. (They are already included in cNarrowCons_FollowsO, so their inclusion here represents duplication.)

2. Show-Invisibles feature has not yet been implemented

3. Dwarf versions of Tai Don aspirated consonants

Dwarf versions of Low Cho (AA8C), High Cho (AA8D), and Low Pho (AA9E) should be made, similar to the Low Co Dwarf (uniAA8A.dwarf). The purpose is to reduce the height of stacked diacritics over these three characters to the same height as those over the Low Co Dwarf.

4. Dwarf version of High NGO

UniAA89.dwarf was created by truncating uniAA89. See Figure 1.

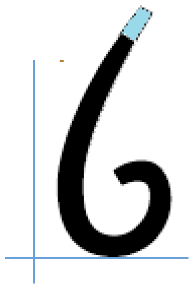

Figure 1: comparing regular and dward version of High Ngo.

black = uniAA89.dwarf
blue = uniAA89

This has the unfortunate effect of moving the tip of the character *and the attachment points* to the left, which aggravates some collision issues.

Would it not be better if the height of the stem (i.e. the left half of the glyph) was scaled so that the tip moved straight down? Besides preserving the natural curvature of the letter, this might also ease some of the tight positioning problems with combining marks. See Figure 2.

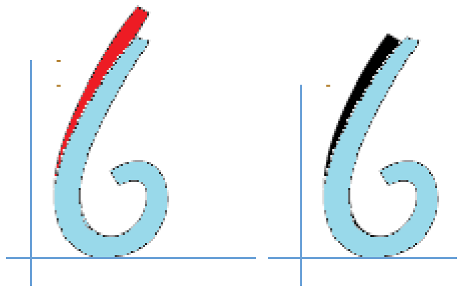

Figure 2: comparing proposed revisionof uniAA89.dwarf with current glyphs

red = uniAA89
black = current uniAA89.dwarf
blue = proposed revision for uniAA89.dwarf

5. Cross-Word Collisions

The current OT and Grapite rules for cross-word collisions all depend on matching a space and kerning the space based on before and after context. The rules work well when they fire (as in WorldPad), but in some applications they never fire (as in Xetex, because of the way Xetex processes the space character). Following is a description of an alternative strategy that does not require matching across spaces, and should be less application dependent.

Conceptual foundation

In the Tai Viet script, the overhang of a glyph will frequently encroach on the space of the glyph following it. If the encroaching glyph is near the end of a word, it may encroach on the following space. (Figure 3)

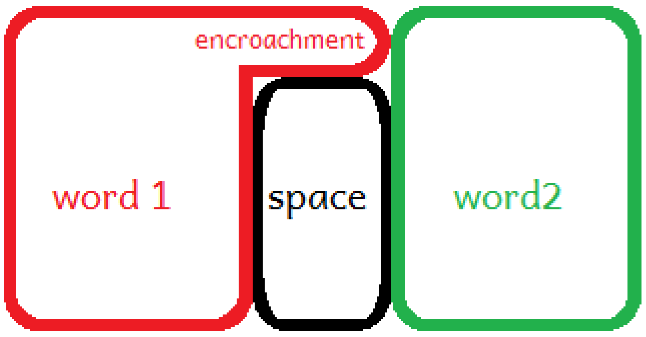

Figure 3: encroachment of Word 1 into the following space.

This is not a problem as long as the encroachment remains over the space character. But when the encroachment extends into the following word, there is the potential for a collision to occur between the two words. (Figure 4)

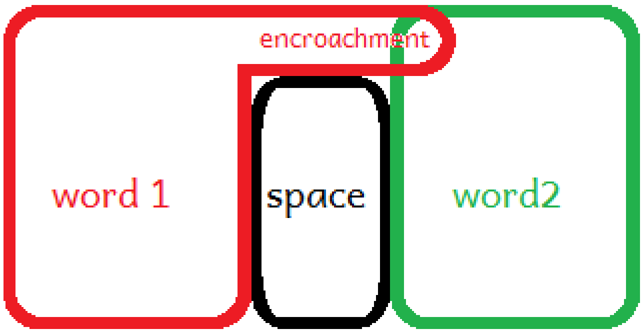

Figure 4: encroachment of Word 1 into Word 2

In the past, we have sought to fix this by expanding the interword spacing. I propose that in the future, we prevent the problem by limiting the amount of encroachment that Word 1 can produce, and guaranteeing a margin in Word 2. (Figure 5)

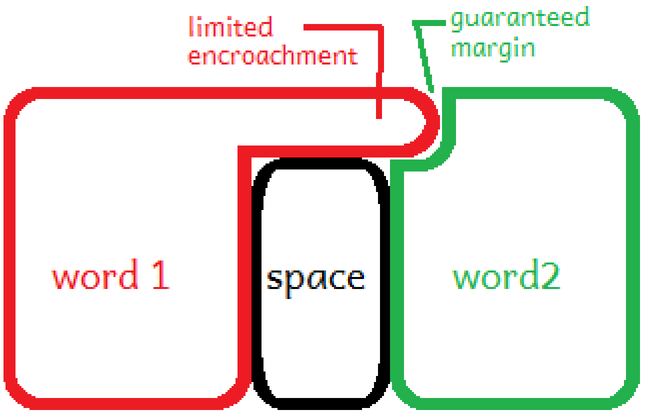

Figure 5: controlling collisions by limiting encroachment and guaranteeing margin

The significant difference here is that the amount of the encroachment is always limited, and the minimum margin is always guaranteed, regardless of whether a collision occurs or not. This should eliminate the need to write rules that include the space character.

Therefore, I propose:

1. Create a set of alternate glyphs for Long-Right-Tail characters (LRT) with clipped tails (e.g. uniXXXX.clipped). These tails will be 100 to 400 units shorter than the default glyphs. Then create lookups/rules which *always* select the clipped tails for LRTs in the context

   LRT > LRT.clipped / _ Combining Vowel? CombiningTone? space

   However, since we are trying to avoid the use of space in our rules, we may need to cast this rule in a negative formate. Perhaps

   LRT > LRT.clipped

   LRT.clipped > LRT / _ Combining Vowel? CombiningTone? class_AllCharExceptSpace

   Or

   LRT > LRT.clipped / _ Combining Vowel? CombiningTone? NOT(class_AllCharExceptSpace)

2. Create lookups/rules which *always* kern either the narrow consonant, the Mai Tho, or the

high combining vowel in the contexts

NarrowConsonant / [noLeftSideVowelOrPunctuation] _ CombiningVowel MaiTho

TallNarrowConsonant / [noLeftSideVowelOrPunctuation] _ HighCombiningVowel

The advantage of this solution over the present one is that none of the rules depend on the problematic matching of a space.

Before implementing this solution, we would need to run some trials to answer some quesions:

- The total adjustment needed to fix a collision is divided between 1 and 2. How much of the adjustment should be handled by 1, and how much by 2?
- When conditions for a cross-word collision exist, both rules 1 and 2 will be fired, and the collision will be corrected. However, there will also be many instances when the conditions for only one of these rules is met, and that rule will be fired even though there is no potential collision. Will this have any noticeable detrimental effect on the overall appearance of a text?

6. Consonant with Low Right Tail + Mai Song (ꫂ)

When Mai Song immediately follows a character with a low tail that curves to the right, their tails may collide or come very close. The tail of the consonant or the Mai Song could be clipped a little, or recontoured. This affects High Co, High Do, Low Tho, and Low Lo.

ꪋꫂ ꪓꫂ ꪖꫂ ꪨꫂ

(Note that this does not affect the current Tai Dam language project. That project is publishing using the combining tones Mai Ek (ꪀ꪿) and Mai Tho (ꪀ꫁) instead of the spacing tones Mai Nueng (ꪀꫀ) and Mai Song.)

7. Vowel position options

Option 1: If Vowel-over-final is retained:

- Word final HighNgo does not need to go to HighNgo.dwarf when a left-side vowel freezes the combining marks over the initial consonant.

Option 2: Simply remove Vowel-over-final feature, and make no other changes

Option 3: Merge the Vowel-over-initial and Vowel-over-final options into a single Vowel-over-mid-word strategy, with the following characteristics:

- Retain the current anchor points, which define the default positions for the combining vowels and tones.
- The presence of a left-side vowel locks the combining marks to their default positions
- Base characters with an upper tail lock the upper combining marks to their default positions. They will also lock a lower vowel to its default position if their is a combining tone present in the data.
- Base characters with a lower-left tail lock a lower vowel, and any combining

tone, to their default positions. But the tone is not locked if there is no lower vowel.

- If the combining marks have not been locked, and if the base character is followed by:
    - Low O as a vowel, or
    - (optional Vowel AA) Final Consonant
- shift the combining marks to the right by 1/2 character.
- NOTES:
    - Must accuratel determine whether follow consonant is a final, or the initial of a new syllable.
    - If both combining vowel and tone are present, both must be moved, or neither.
    - Move can be made by using an alternate set of APs, or by kerning commands.

Advantages of Option 3:

- Most closely resembles hand writing.
- Uses a "where there's most room" approach, which reduces opportunities for collisions.

8. Dwarf glyph for AABB ꪻ

AABB collides with the tail of the preceding consonant in the sequences (AA9C | AAA4) + (OptionalCombiningMarks) + AABB (ꪜꪻ or ꪤꪻ). To reduce the extreme amount of kerning needed to fix the collision, AABB was moved both right and *down*. It would be better if a dwarf version of AABB could be substituted, so that only a modest amount of kerning to the right would be needed.