Smart Fonts and Keyboards
Keyman and the Brave New World

Martin Hosken
SIL Non-Roman Script Initiative

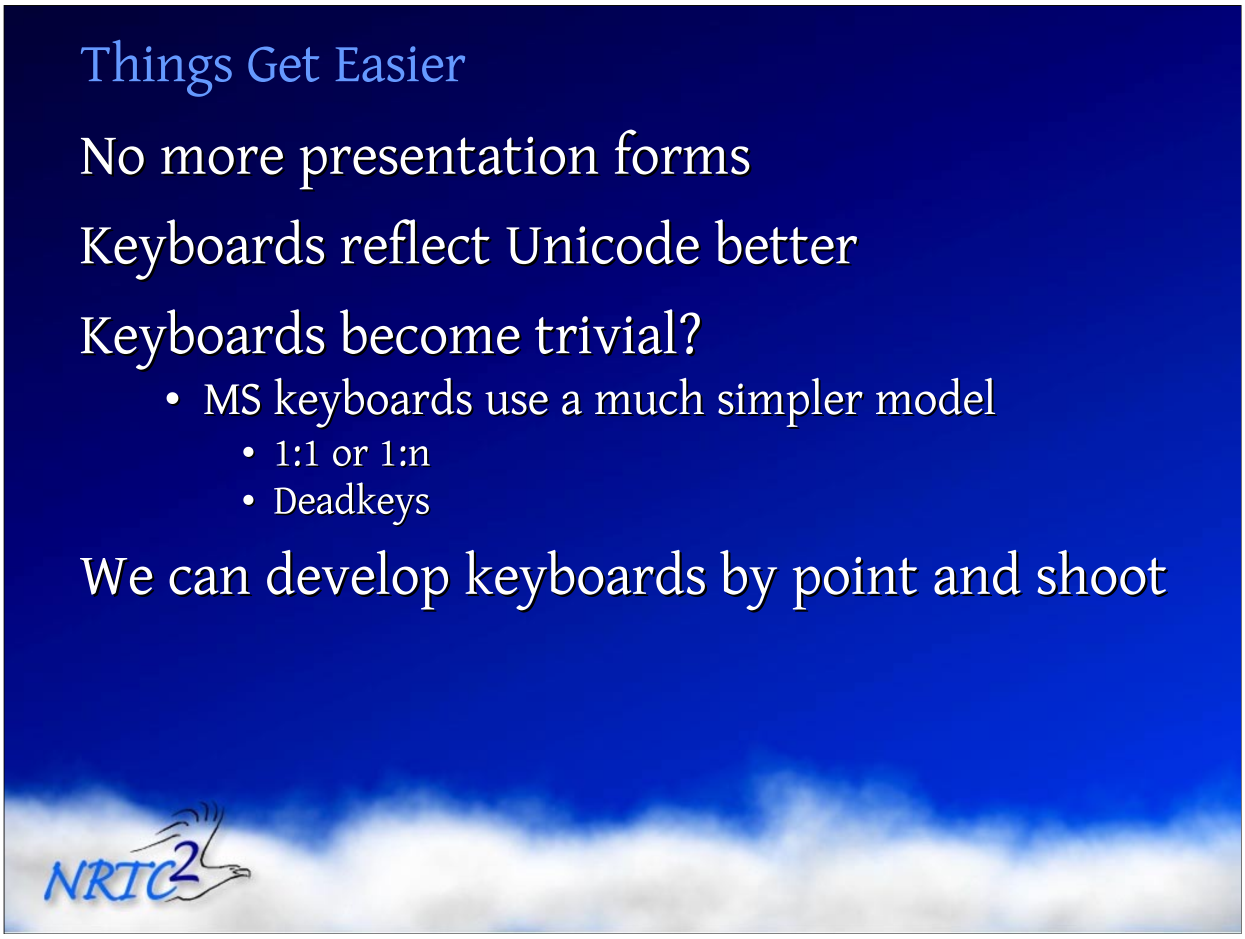
NRTC2

This presentation requires the following fonts:

SIL NRTC Presentation

Courier New

Inle Academy



Things Get Easier

No more presentation forms

Keyboards reflect Unicode better

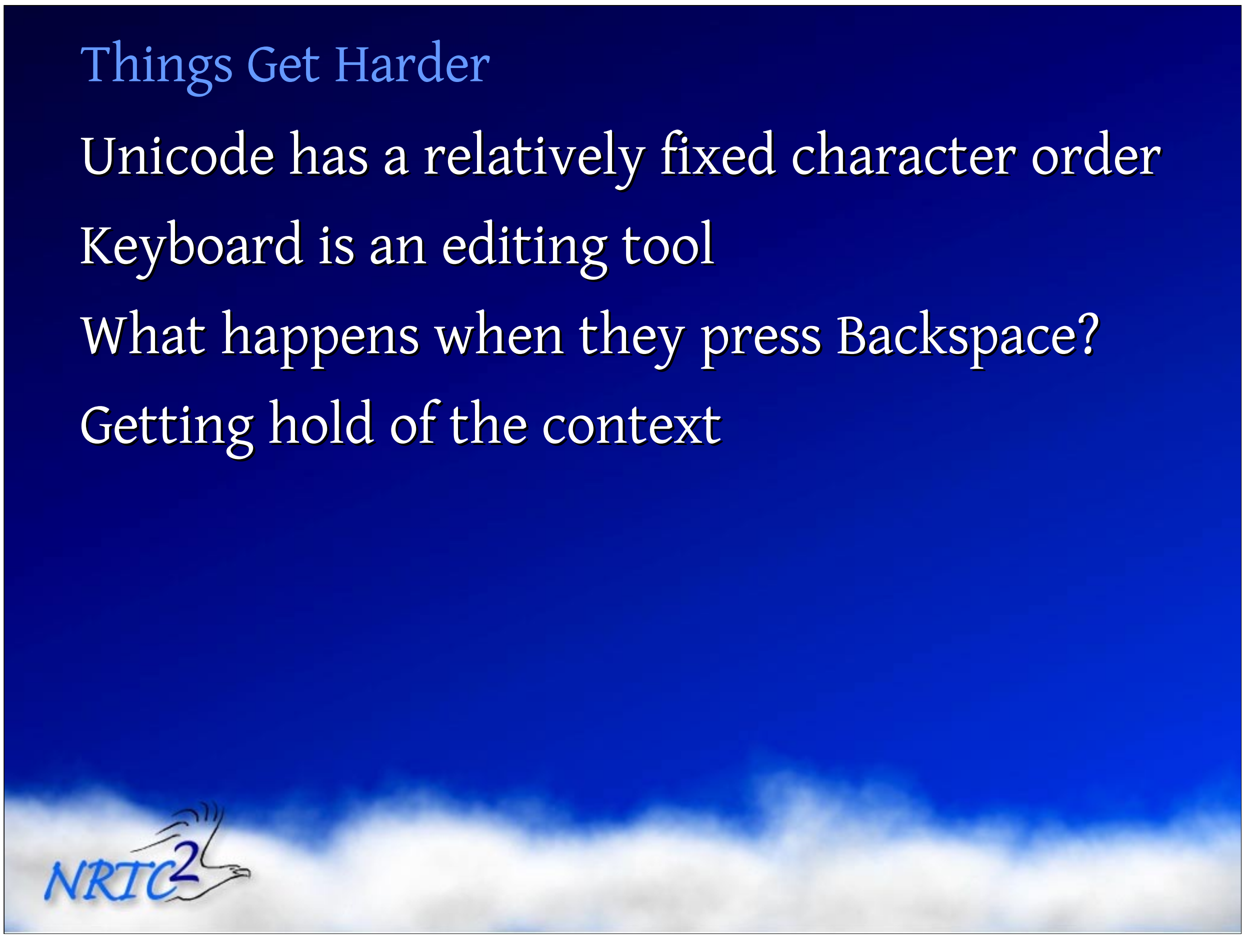
Keyboards become trivial?

- MS keyboards use a much simpler model
  - 1:1 or 1:n
  - Deadkeys

We can develop keyboards by point and shoot

Things get easier because we don't have to use the keyboard to do the rendering by substitution, that has now gone into the font.

Don't things now become trivial with keyboards becoming, basically, a 1:1 mapping between a key and a Unicode value. There is no need for contexts now right?



Things Get Harder

Unicode has a relatively fixed character order

Keyboard is an editing tool

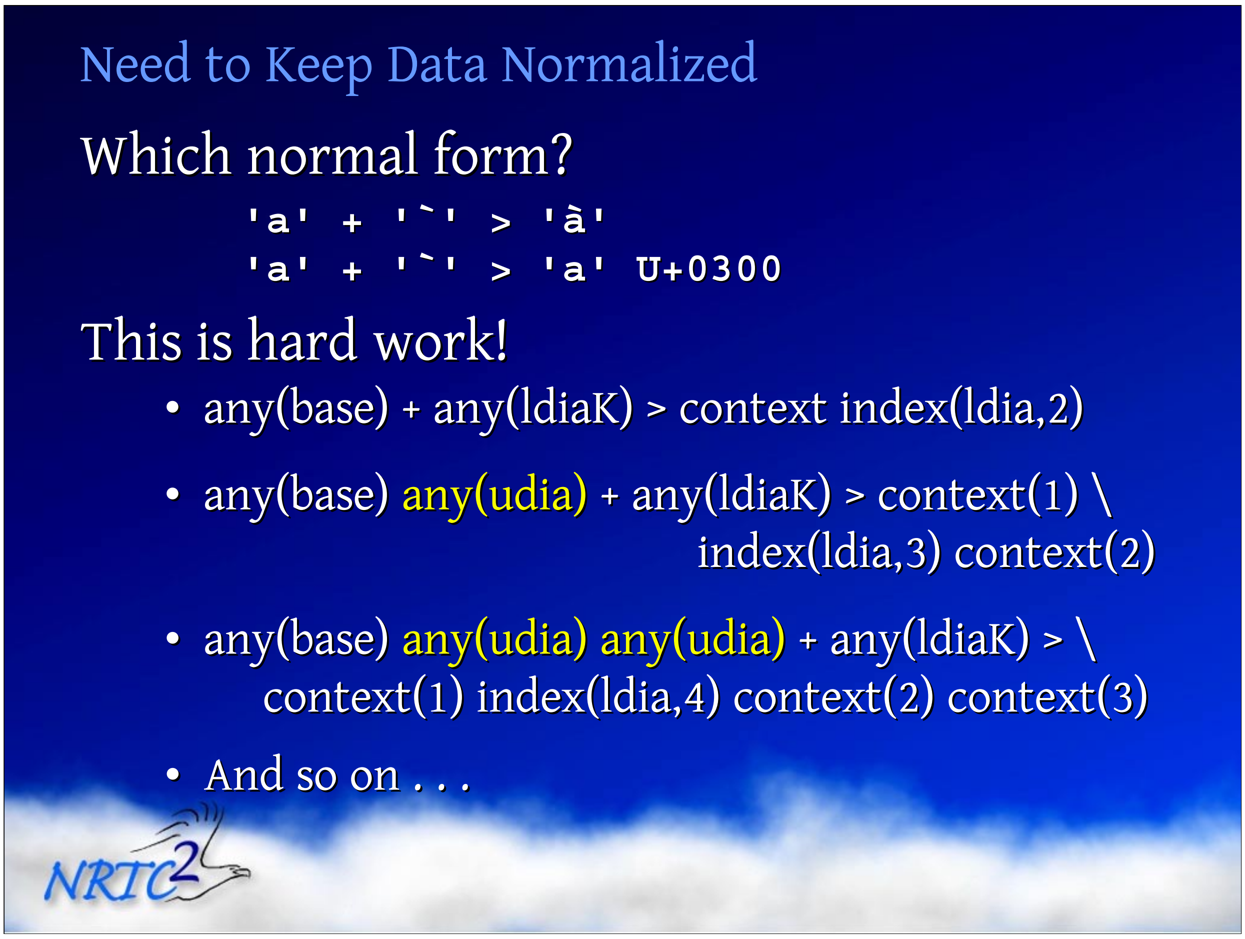
What happens when they press Backspace?

Getting hold of the context

For some, this is the case. Keyboard definitions become nearly trivial. If you don’t have any diacritics and are working with a simple script, then you are home and dry.

Well, not really.

There are a number of issues that we need to think about when it comes to working with Unicode.



Need to Keep Data Normalized

Which normal form?

```
'a' + '`' > 'à'
'a' + '`' > 'a' U+0300
```

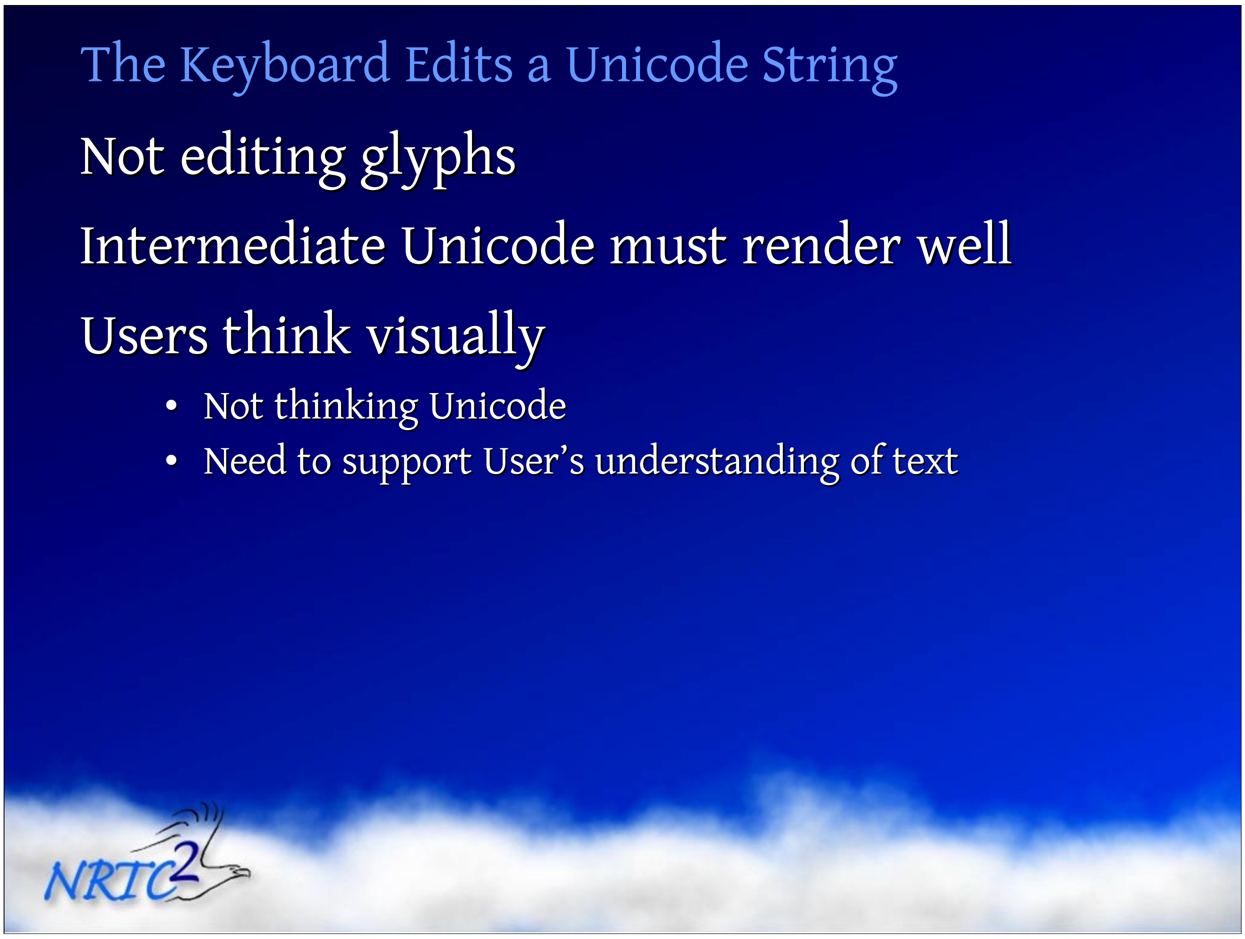
This is hard work!

- any(base) + any(ldiaK) > context index(ldia,2)
- any(base) any(udia) + any(ldiaK) > context(1) \
  index(ldia,3) context(2)
- any(base) any(udia) any(udia) + any(ldiaK) > \
  context(1) index(ldia,4) context(2) context(3)
- And so on . . .

The first need is that applications like their data normalized.

But which normal form? Do we need different keyboard layouts for different normal forms? Software developers. What are you expecting from the keyboard when it comes to editing? Can the keyboard pass in strings that have to be normalized?

The keyboard needs to at least keep things in canonical order. But this is hard work and tedious. Will Keyman do this for us and allow a mechanism for an application to say which normal form they want or for Keyman to tell the application what normal form it is getting?



The Keyboard Edits a Unicode String

Not editing glyphs

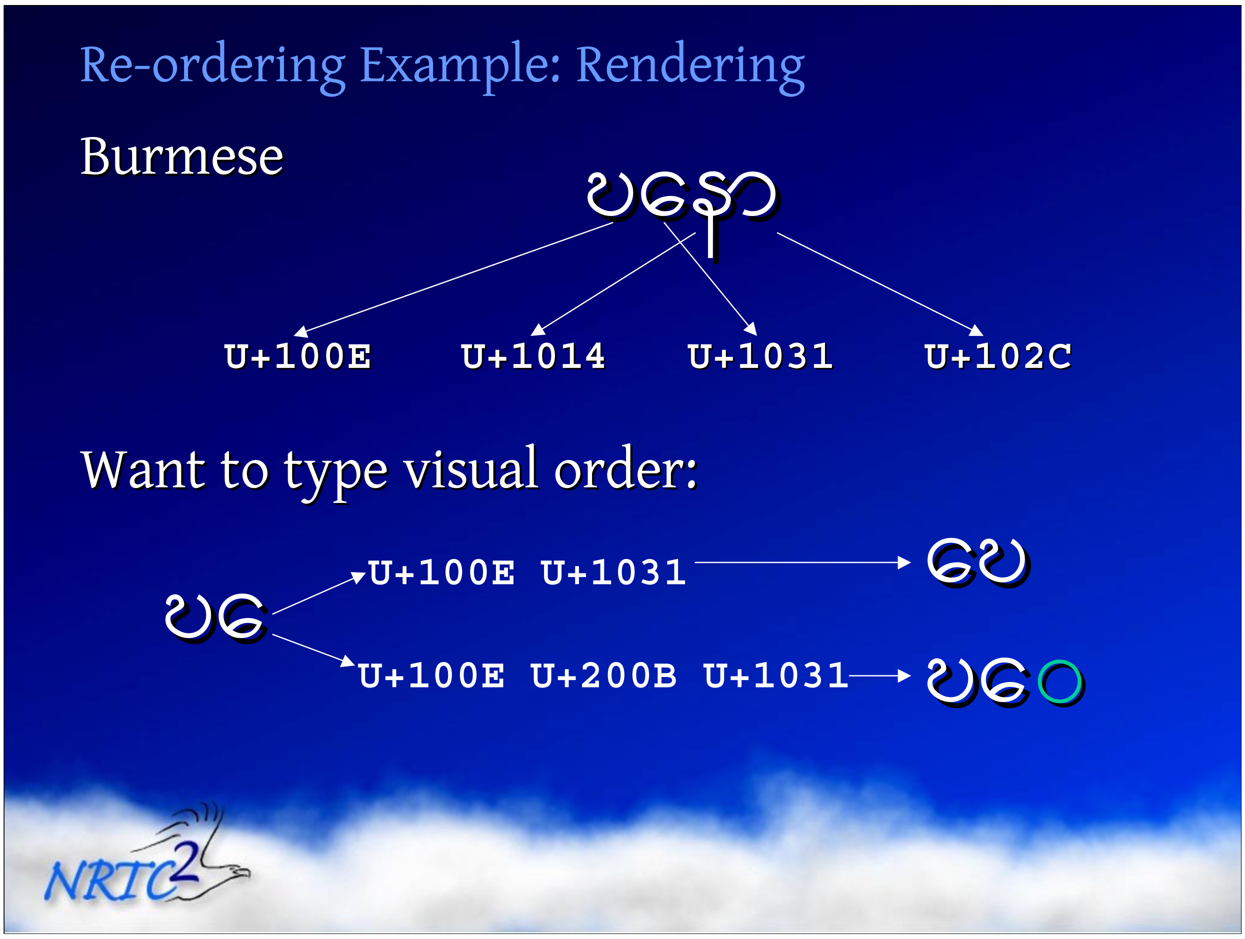
Intermediate Unicode must render well

Users think visually

- Not thinking Unicode
- Need to support User's understanding of text

NRTC2

The keyboard is not editing the glyph stream, it is editing the underlying Unicode text. But while you are editing away, you need the renderer to be able to do something sensible with the intermediate form of the string as you are part way through typing you masterpiece. In most cases, this is not a problem, but there are some situations where things can get hairy.



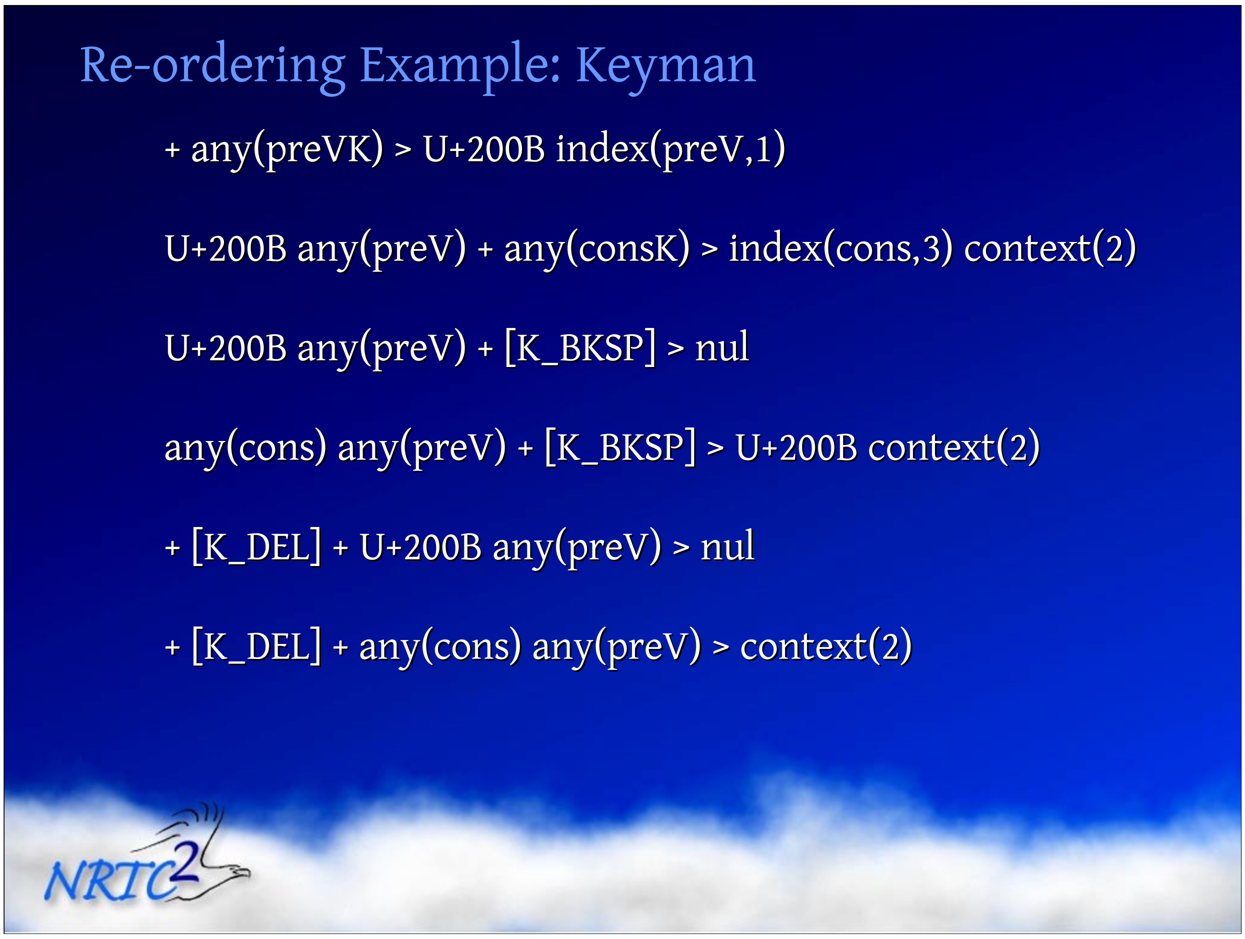
Re-ordering Example: Rendering

Burmese

Want to type visual order:

Consider this example from Burmese (ddhano). The second and fourth letters together form a vowel 'o'. And in Unicode the vowel comes after the consonant. So the two parts to the vowel are stored after the consonant. So the underlying store has two consonants (the first and 3rd characters) and then 2 vowels (chars 2 & 4). Now, most people like to type things in visual order rather than underlying order. Thus they want to type the pre-vowel (U+1031) before the following consonant. Thus they type the first letter (U+100E) and then the pre-vowel, but what happens in the renderer?

Instead we need to insert a blank for the vowel to interact with. Some people like to render such situations using a dotted circle to show very clearly that something is up. That is optional in this case, but we need to mark what's going on.



Re-ordering Example: Keyman

```
+ any(preVK) > U+200B index(preV,1)

U+200B any(preV) + any(consK) > index(cons,3) context(2)

U+200B any(preV) + [K_BKSP] > nul

any(cons) any(preV) + [K_BKSP] > U+200B context(2)

+ [K_DEL] + U+200B any(preV) > nul

+ [K_DEL] + any(cons) any(preV) > context(2)
```

The first line handles pressing our pre-vowel. We insert a ZWSP when we type the pre-vowel so that the pre-vowel won't suddenly try and render before the previous character.

The second line handles pressing a subsequent consonant. We need to re-order it before the pre-vowel and delete the ZWSP

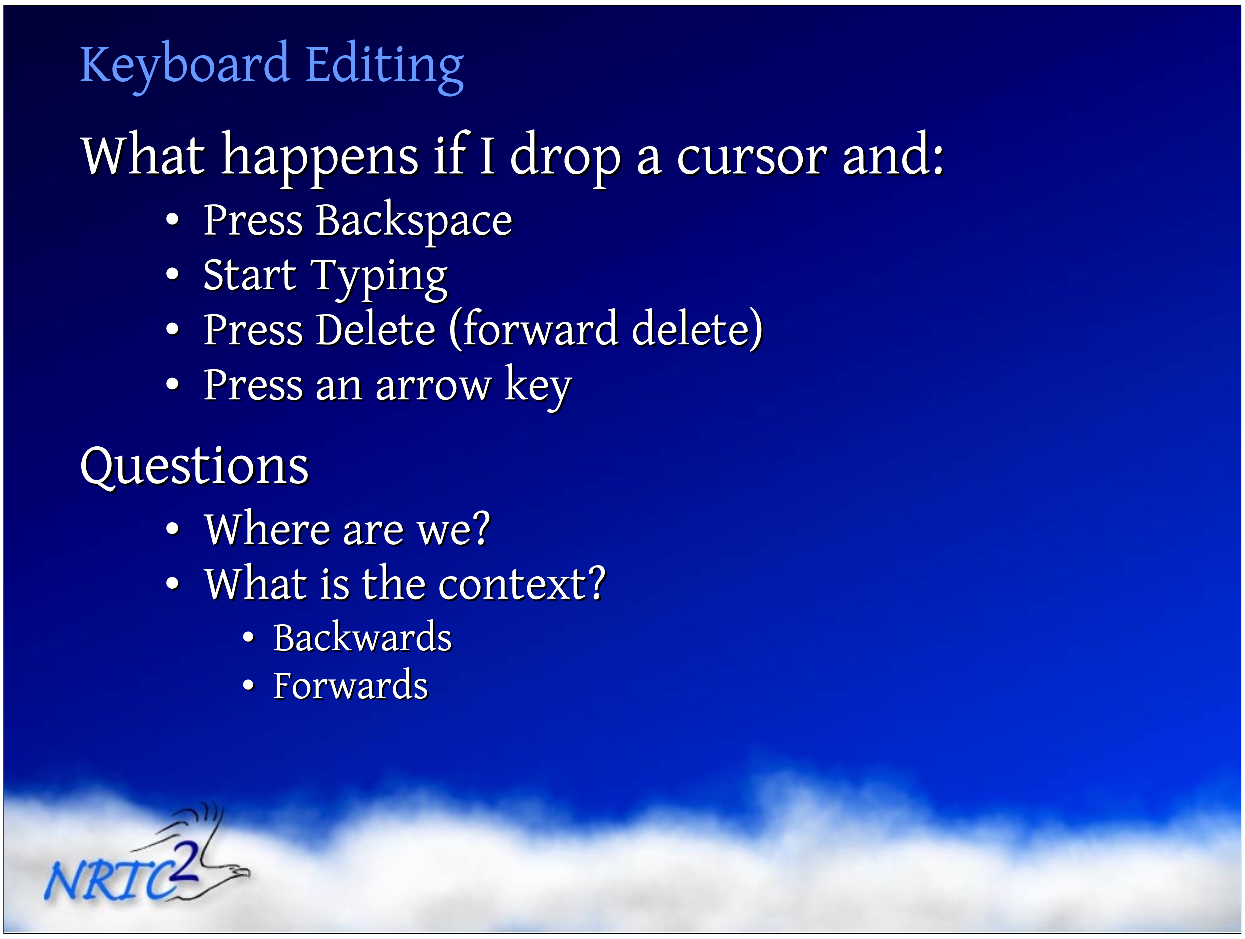
The third line addresses pressing backspace and the need to delete the pre-vowel and the ZWSP as a unit.

The fourth line handles pressing backspace following a consonant with a pre-vowel. Here we need to make sure we end up with the ZWSP and the pre-vowel.

The fifth line is odd, it is my attempt at some type of forward context where the keyboard looks ahead when I press the DEL key to delete both the ZWSP and the pre-vowel as a unit.

The sixth line handles pressing del before a pre-vowel which has a following consonant.

As you can see. Things can get pretty hairy when the keyboard doesn't fit well with Unicode ordering. But this is what is needed. The keyboard must deal with things in terms of the underlying Unicode and as you design your keyboard you need ot be thinking all the time about how your intermediate strings are going to be rendered.



Keyboard Editing

What happens if I drop a cursor and:

- Press Backspace
- Start Typing
- Press Delete (forward delete)
- Press an arrow key

Questions

- Where are we?
- What is the context?
  - Backwards
  - Forwards

NRTC2

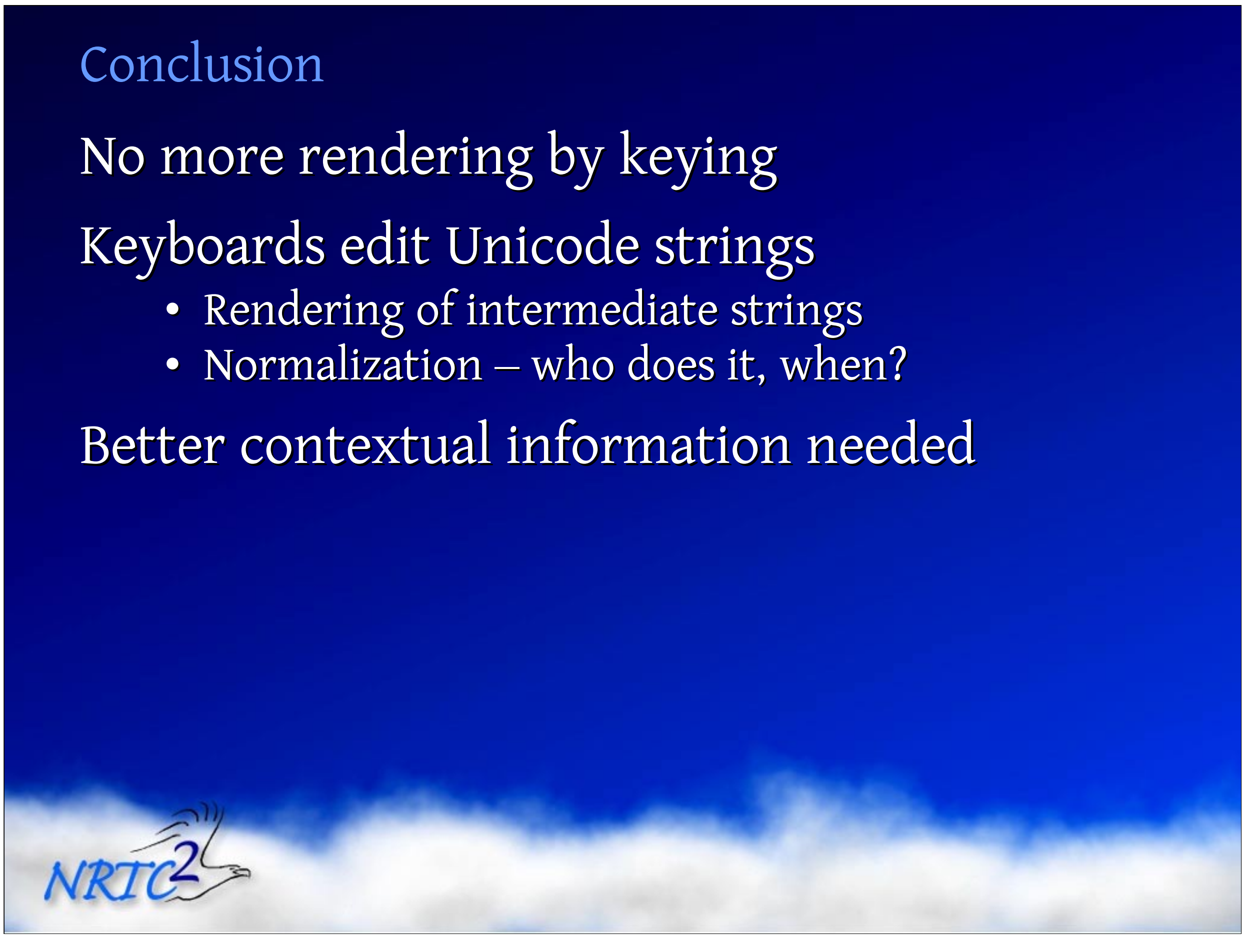
Here are some questions that you need to consider as you develop your keyboard.

When you press an arrow key, what happens?

1. Keyman moves its context
2. The application asks the rendering subsystem where to move the cursor too, or else just moves it one code in the underlying string.

Are these two in synch? Arrow key control is application specific. It would be better for Keyman to be able to interrogate the application and find out where it is. The problem is that if you have to write rules in Keyman to handle the arrow key, then your keyboard definition becomes application specific, and this is no fun.

Of course MS gets around this by not having any contextual keyboards ☺



Conclusion

No more rendering by keying

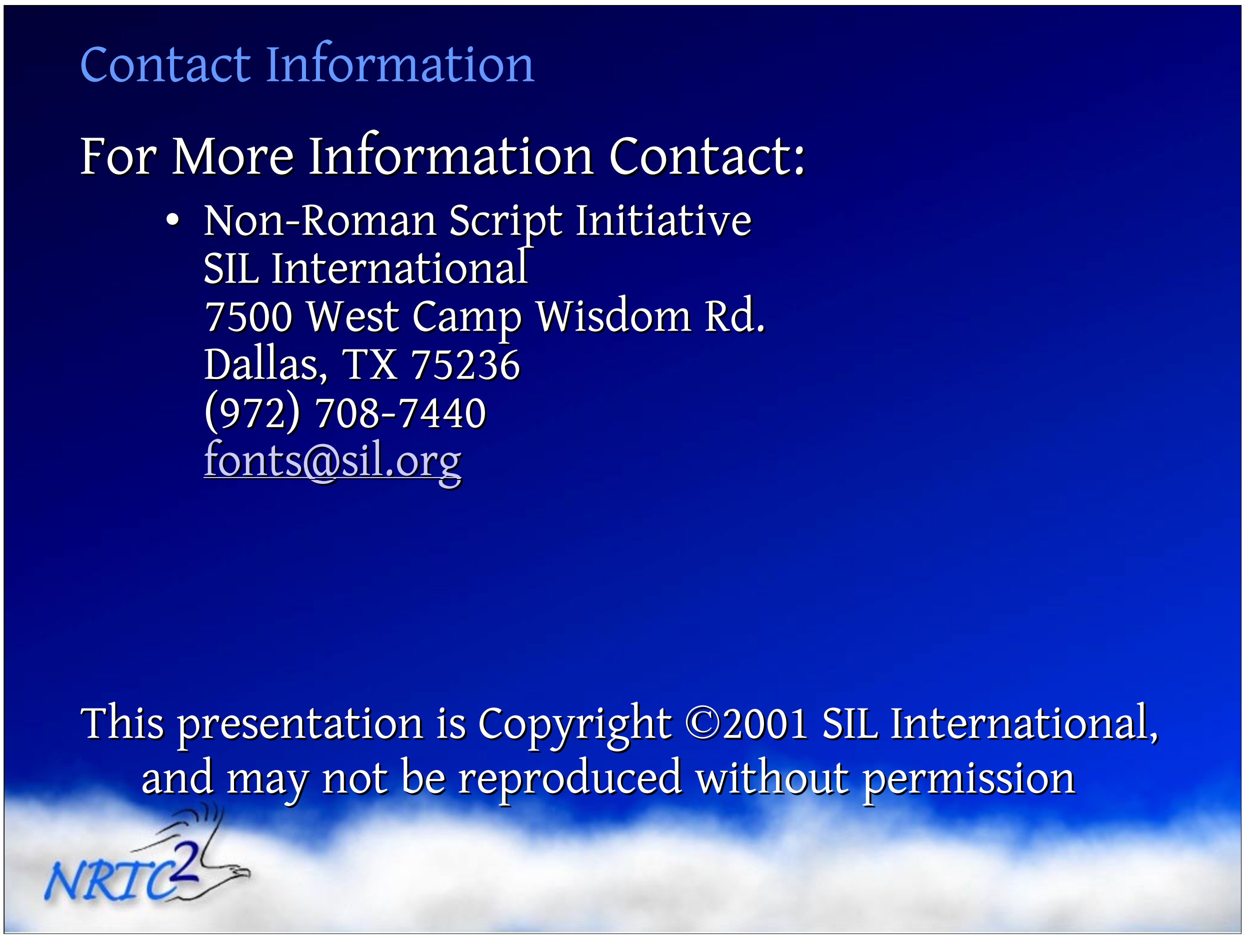
Keyboards edit Unicode strings

- Rendering of intermediate strings
- Normalization – who does it, when?

Better contextual information needed

Questions

Questions and comments?



Contact Information

For More Information Contact:

- Non-Roman Script Initiative
  SIL International
  7500 West Camp Wisdom Rd.
  Dallas, TX 75236
  (972) 708-7440
  fonts@sil.org