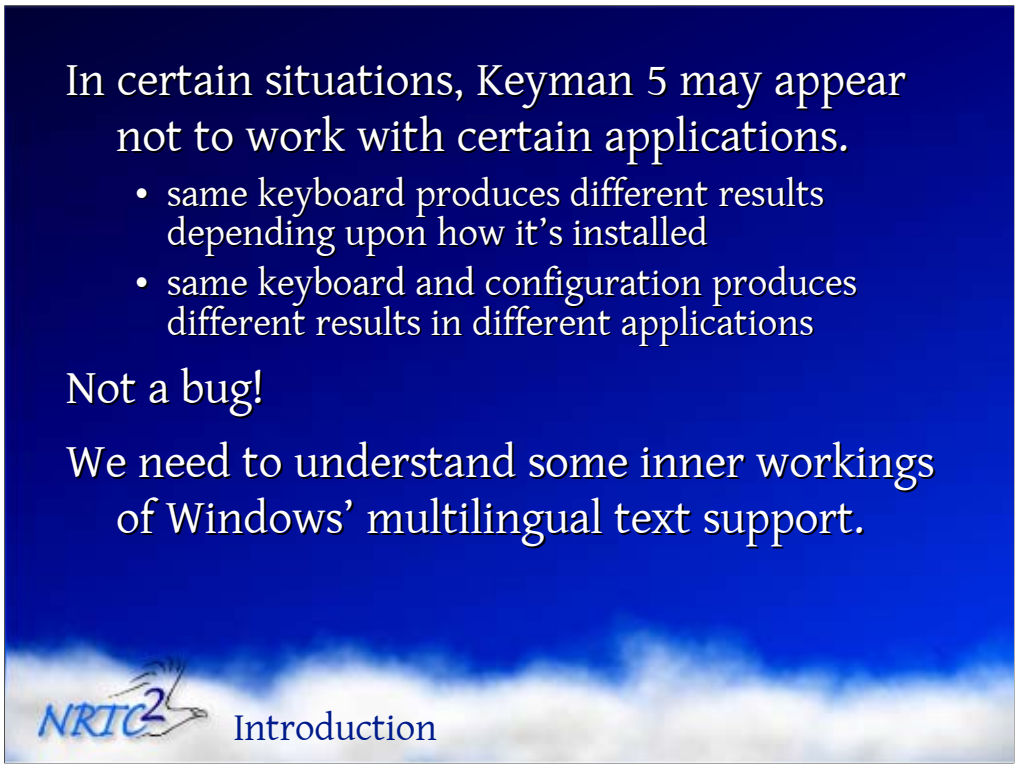
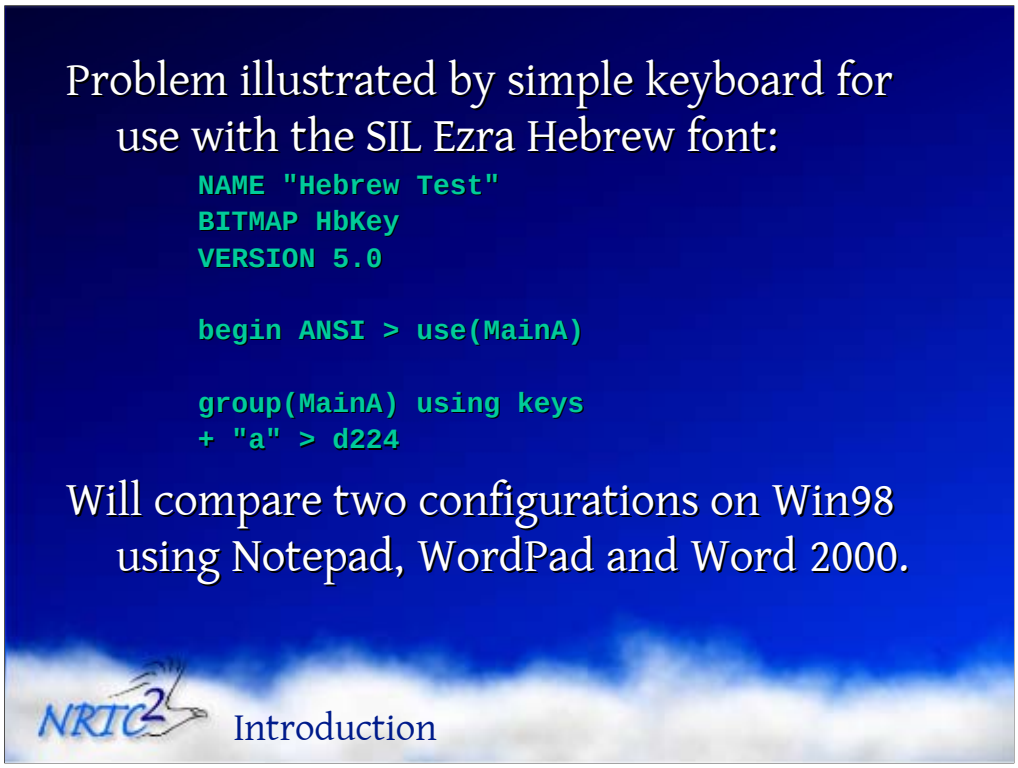
Keyman, LANGIDS & Codepages
Interactions you may not expect

Peter Constable
SIL Non-Roman Script Initiative



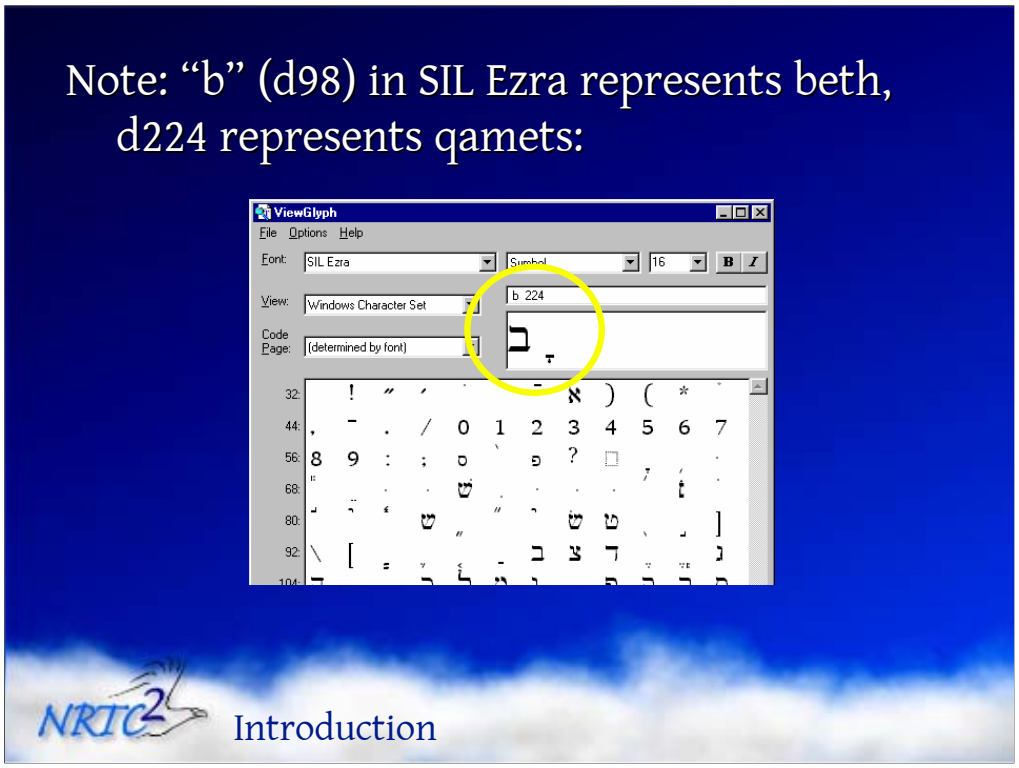
There are certain interactions between Windows multilingual infrastructure and Keyman 5 that may lead people to think Keyman has bugs. Specifically, it may appear that Keyman does not work with certain applications. With certain applications, a Keyman keyboard may produce different results depending upon how the user has installed that keyboard. It's also possible to have a Keyman keyboard on a single machine produce different results in different applications.

This is not a bug! It is actually one of the potential by-products of allowing people to add customised multilingual capabilities but without ensuring that all of the inter-relating pieces are in place. To avoid these problems, we need to understand some of the inner workings of Windows' multilingual text support.

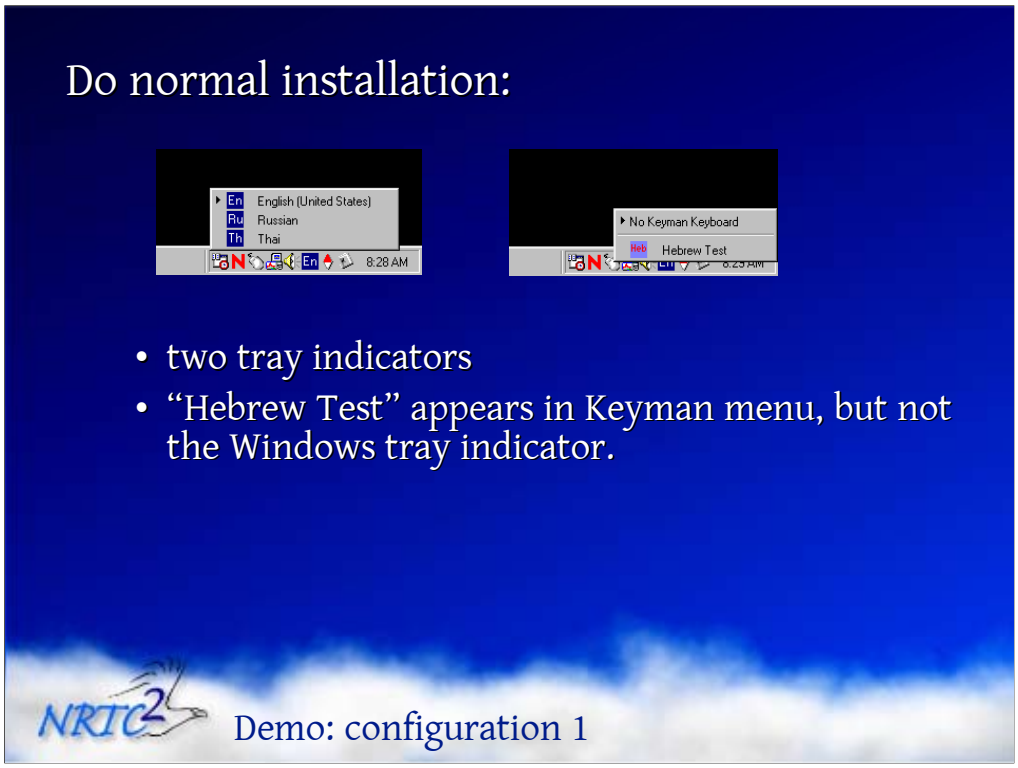


The problem can be illustrated with a very simple keyboard. I will use one that is an abridged version of a keyboard that would be used with the SIL Ezra Hebrew font. In this keyboard, the “a” key is used to generate the character code d224 (0xE0).

To demonstrate the problem, we will compare two configurations of this keyboard on Windows 98 using Notepad, WordPad and Word 2000.

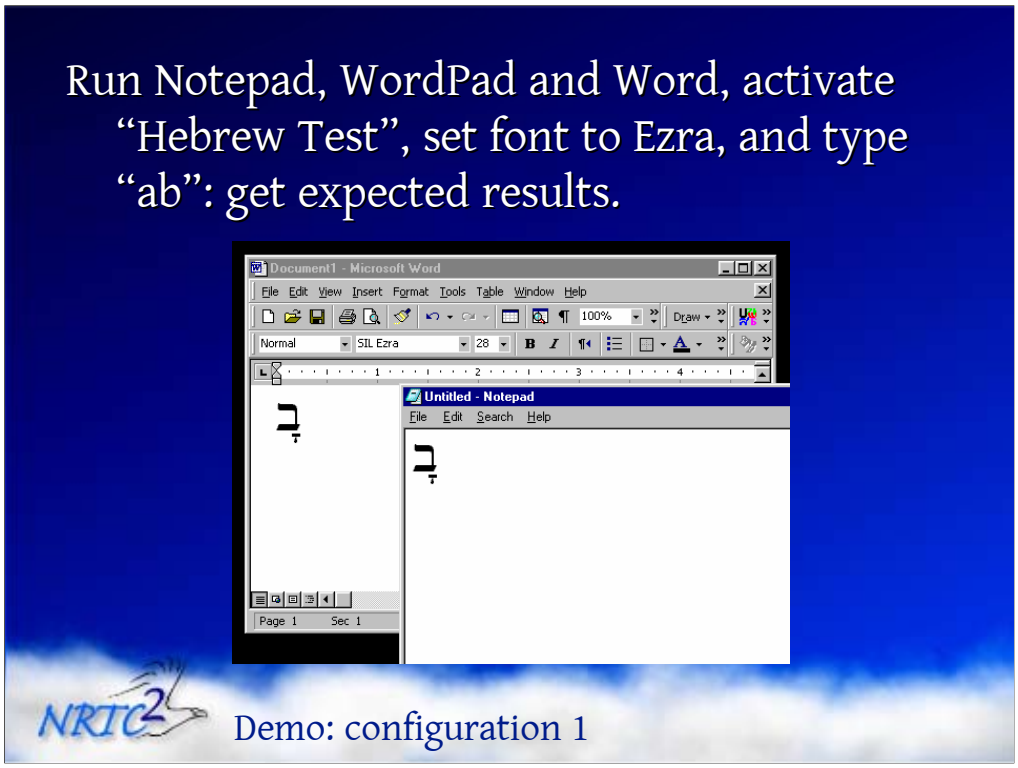


We will be using keystroke sequences involving the “a” and “b” keys. The “b” key is not remapped in our keyboard, and so will generate an ANSI “b” character (d98 = 0x62). In the encoding of the SIL Ezra font, this represents the Hebrew letter beth, “ב” As mentioned above, the “a” key will generate the character code d224 (0xE0). In the Ezra encoding, this represents the Hebrew vowel qamets “ָ”. We will be using the sequence < “a”, “b” >, which should generate < qamets, beth > (which is the correct ordering for a visually-ordered encoding as required by Ezra).



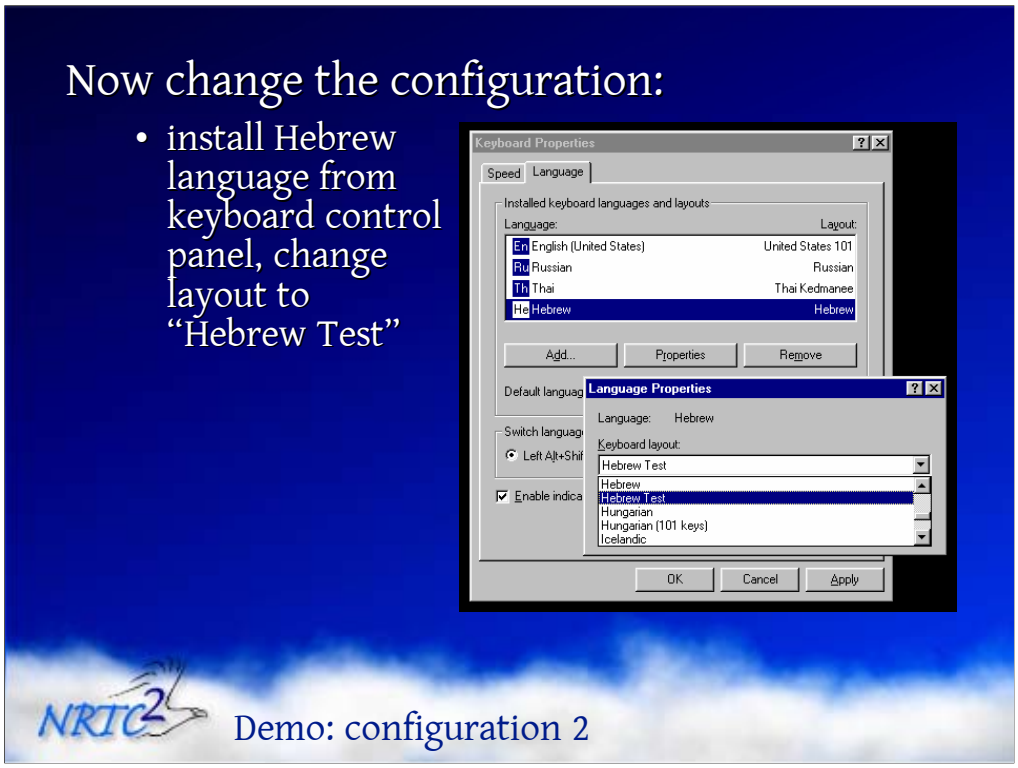
The first configuration we will consider is a normal keyboard installation, i.e. the default without the user taking any special steps.

When the keyboard is compiled and installed, we will see two tray indicators for input methods: Windows' indicator (internat.exe, shown on the left), and Keyman's. Our "Hebrew Test" keyboard appears in the Keyman menu, but nothing changes in the menu for the Windows tray indicator.



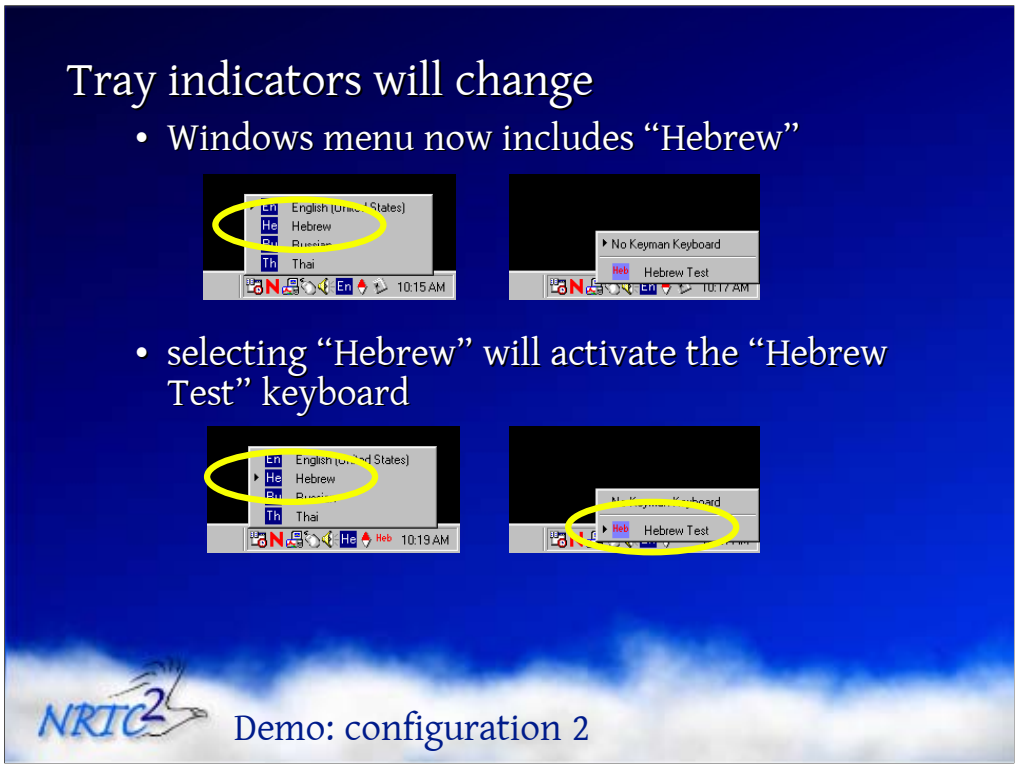
Now, if we run Notepad, WordPad or Word, activate the "Hebrew Test" keyboard using Keyman's tray indicator menu, set the font to SIL Ezra, and then type "ab", we will get the expected results: < qamets, beth >. This first configuration works as desired.

Note: this qamets glyph is not the positional variant that is preferred for beth. That is irrelevant for the purposes of this exercise, however. (I mainly chose d224 because it made it easy for me to detect the unexpected behaviour we will see shortly.)

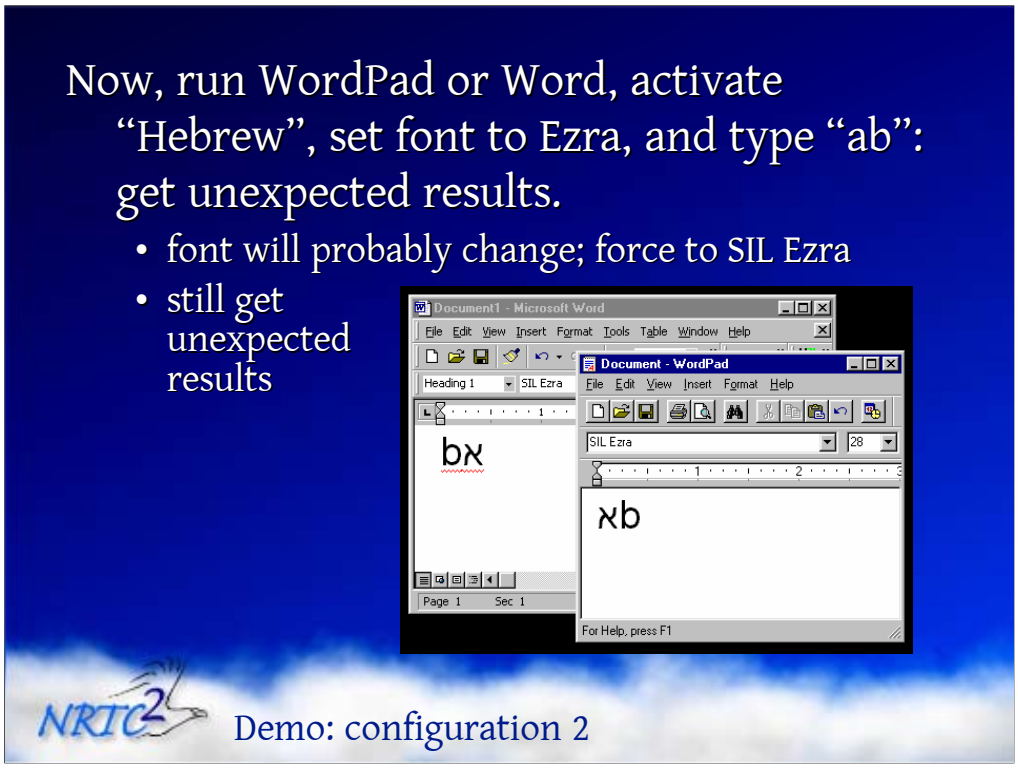


The second configuration involves an additional installation step that users may take if they want to associate the keyboard with a specific language.

To do this for our Hebrew keyboard, we go to the Keyboard Properties control panel and add the Hebrew language to the list of installed keyboard languages. Then we modify the properties for Hebrew to set the layout to our "Hebrew Test" keyboard.



This will bring about some changes in the menus shown by the tray indicators. First, the Windows tray indicator menu will now include “Hebrew”. Secondly, if we select “Hebrew” in the Windows menu, that will have the effect of activating the “Hebrew Test” keyboard, and it will appear as selected in the Keyman menu.

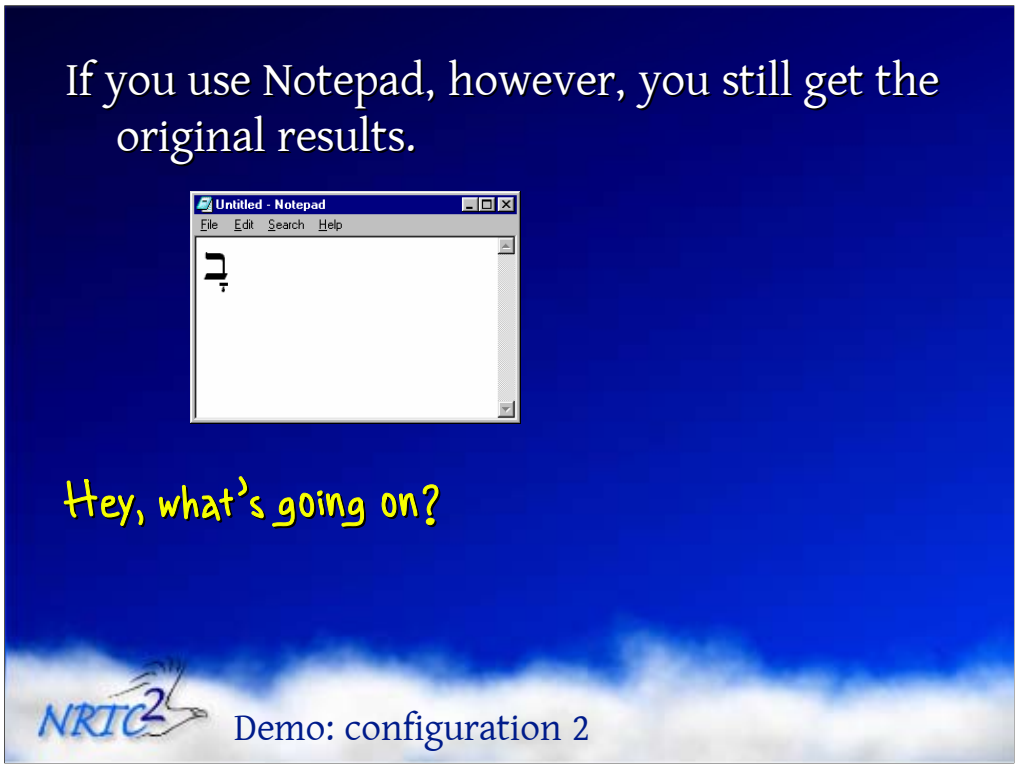


Now, we want to test this configuration in the three applications. First, we run WordPad or Word, activate "Hebrew" in the Windows tray indicator menu, set the font to Ezra, and type "ab". Immediately we see some odd results. For one thing, the font will change (I expect this would happen on all installations, though I am not certain about this with regard to WordPad.) With a bit of coercion using the Format | Font dialog, we may be able to get the font set back to Ezra. Even if we do, though, we can't avoid the second result: we don't get the characters that were expected.

The exact characters that appear may vary depending upon the particular system, application version and application settings. The "b" keystroke may show up as a "b" in a font other than Ezra, or it may show up as the Ezra beth as expected. The only issue here is the way that some applications manipulate fonts; the character code will always be d98 as expected.

The "a" keystroke is the interesting part. What will appear is either an empty box glyph, or an aleph, and if an aleph appears it will not be from the SIL Ezra font.

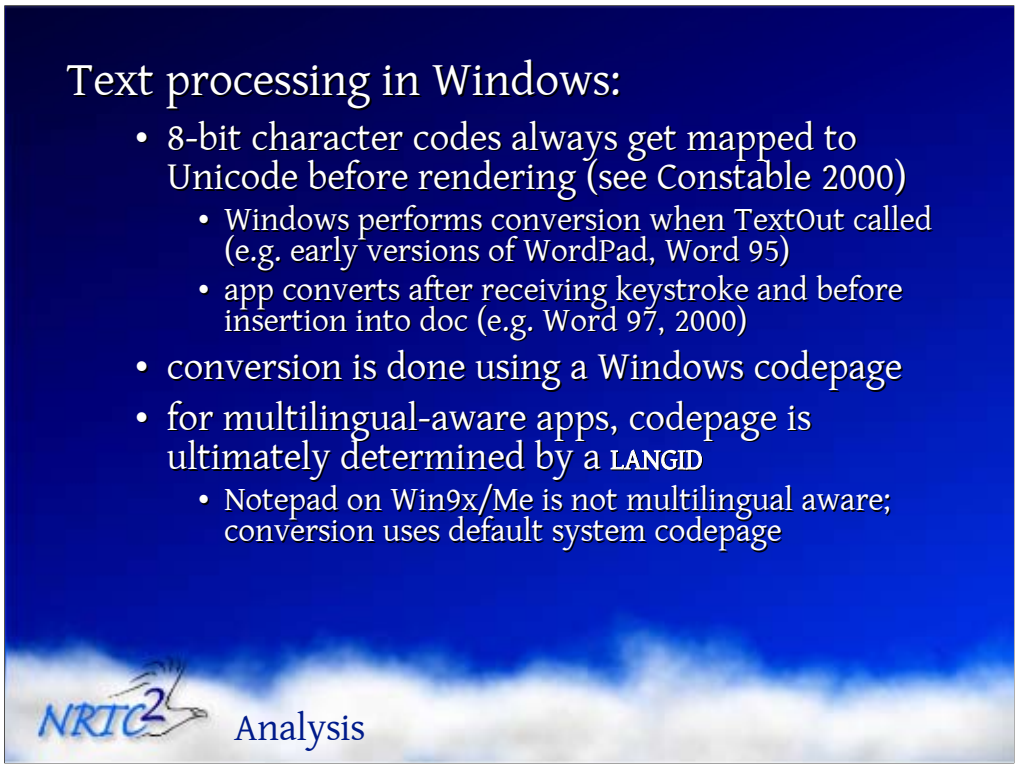
Another interesting issue is that even though we always use the keystroke sequence "ab", we may see characters appearing in different orders in different applications.



There is a lot of weird behaviour happening here, but there is more: we have not yet looked at what happens in Notepad. If we repeat the exercise in Notepad, everything works as before.

At this point, you should be wondering, "What's going on?" The same keyboard is producing different data in the same applications (Word and WordPad) depending upon how it was installed, and a single configuration of the keyboard is producing different data in different applications. We are also seeing some issues with font selections and ordering.

Obviously, this kind of behaviour would be confusing for a user and may seem like a bug in Keyman—these things certainly never happen with keyboards provided by Windows.



To understand what is happening, we need to look into some of the internal workings of multilingual text processing in Windows. These are discussed in detail in Constable 2000. We won't look at all of the details here. Please consult that paper for further information.

The key facts are these:

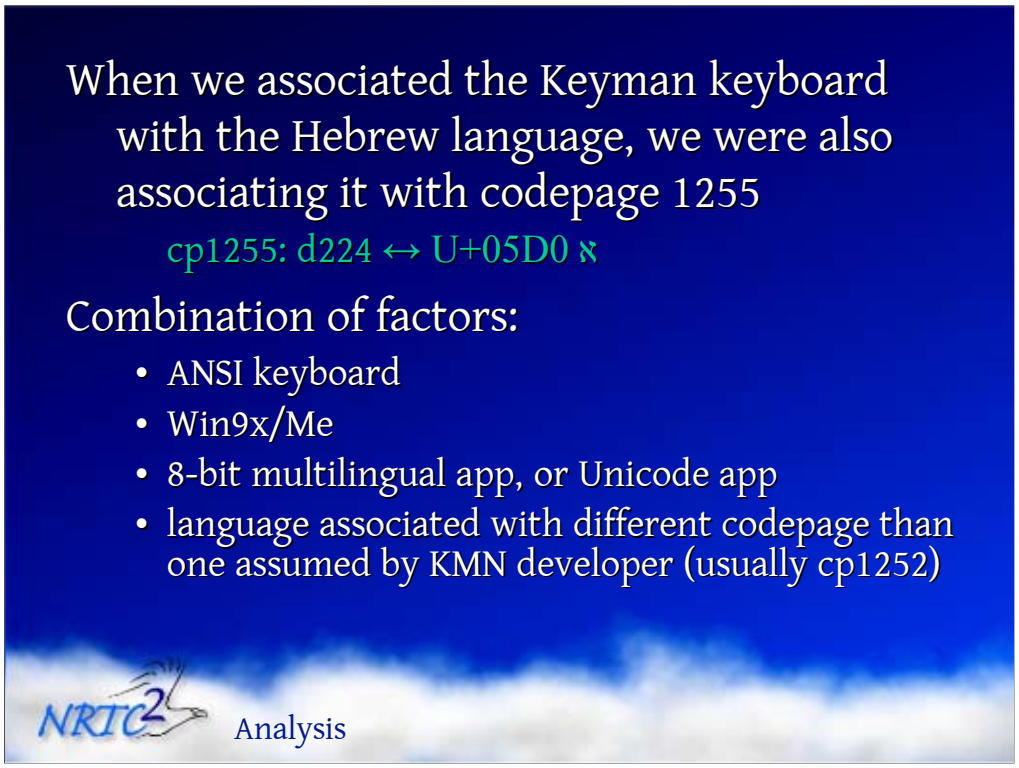
• First, 8-bit character codes always get mapped to Unicode at some point before rendering occurs.

This is necessary because Windows fonts use Unicode character codes. The conversion can take place at one of two points. It can be done automatically by Windows when the application calls TextOut. This would occur with multilingual-capable applications that use 8-bit text, such as early versions of WordPad or Word 95. The second point at which it can occur is within the application immediately after the keystroke has been processed. This would occur with applications that store Unicode text, such as Word 97 or Word 2000.

• Secondly, the conversion is done using a Windows codepage.

• Thirdly, for multilingual-aware applications, the choice of codepage is ultimately determined by a language identifier (LANGID).

Note that Notepad on Windows 9x/Me is not multilingual aware. For Notepad, the conversions are always done using the default system codepage.



It should be fairly clear by now that the key difference occurred when we associated the "Hebrew Test" keyboard with the Hebrew language. When users do this, they are not aware that, in the process, they are also associating the keyboard with a specific Windows codepage, codepage 1255.

The details in our particular tests are that codepage 1255 maps character code d224 to the Unicode character U+05D0 HEBREW LETTER ALEPH "א". If this is displayed with SIL Ezra, an empty box will appear since the font does not support that character. Some applications detect that a Unicode Hebrew font is needed, and will switch the font to one that supports the Hebrew range of Unicode. Some will even display text with such a font even though the text is formatted using Ezra. Some applications will also detect the use of Hebrew script and apply right-to-left ordering. What happens as far as fonts and directionality are concerned will vary from one application to another. The conversion will occur in every case, however.

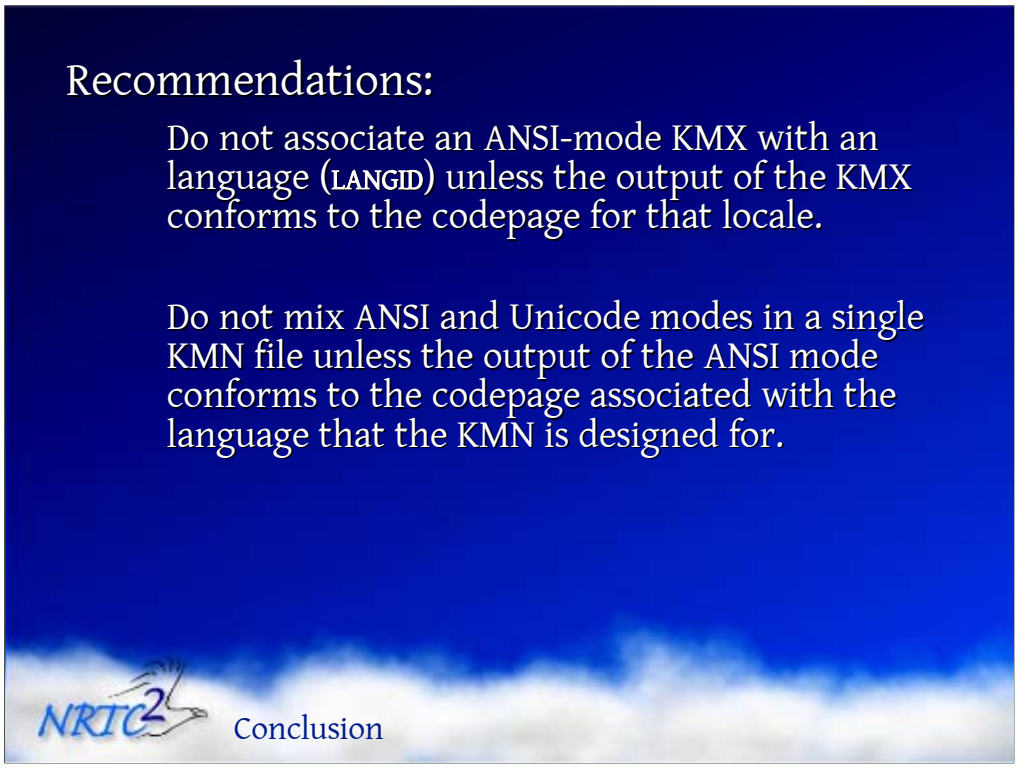
The following combination of factors will result in this problem:

• an ANSI mode Keyman keyboard

• Windows 9x/Me

• an application that makes use of Win32 infrastructure for handling multilingual text, whether it stores text as 8-bit or as Unicode

• a language that is associated with a different codepage than the one that was assumed (perhaps unknowingly) by the keyboard developer for use with particular fonts (usually this is codepage 1252)



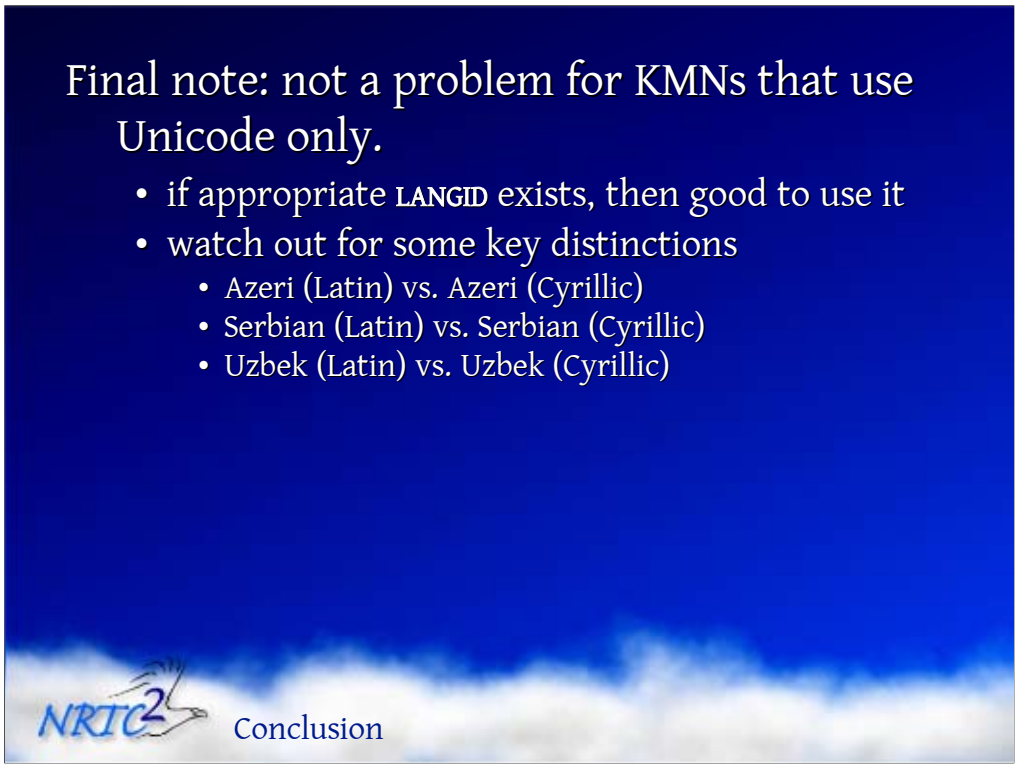
There are two recommendations to be made in view of all this:

• First, users should not associate an ANSI-mode Keyman keyboard with a language (technically, a LANGID) unless the output generated by that keyboard matches exactly the codepage that is associated with that language.

If necessary, keyboards that produce a non-standard encoding should include release notes warning users not to associate the keyboard with a Windows LANGID. This is not likely to be necessary for keyboards that are designed for minority languages that are not supported in Windows, unless there is some likelihood that users might be inclined to associate the keyboard with a language that is supported and that uses a codepage other than codepage 1252 (e.g. Hebrew or Greek).

• Secondly, Keyman keyboards should not mix ANSI and Unicode modes within a single file unless the output generated by the ANSI mode matches exactly a codepage that is associated with the language that the keyboard is being designed for.

If ANSI and Unicode modes are combined but the ANSI mode does not conform to an appropriate codepage, then the keyboard will produce incompatible data in different applications. Thus, for example, it would not be appropriate to create a keyboard for Lao that combines a Unicode mode and an ANSI mode for some custom Lao encoding. On the other hand, it would not be a problem to create a keyboard for Thai that combines a Unicode mode with an ANSI mode that conforms to the Thai standard encoding (TIS 620, which matches codepage 874).



It should be noted that this issue applies only to ANSI-mode keyboards. It is not a problem for Keyman keyboards that use only Unicode modes. There is absolutely no reason for users not to associate such keyboards with a particular LANGID, provided they use the correct LANGID. (For example, it would be important to distinguish "Azeri (Latin)" from "Azeri (Cyrillic)"—getting the wrong one in such a case could create problems.)

Indeed, if there is an appropriate LANGID, they probably should associate the keyboard with it in order to get the best support from applications. Of course, most situations in which Keyman is used involve minority languages for which there is no Microsoft-defined LANGID.

References:

Constable, Peter. 2000. “Understanding multilingual software on MS Windows: The answer to the ultimate question of fonts, keyboards and everything.” Ms.

For More Information Contact:

- Non-Roman Script Initiative
  SIL International
  7500 West Camp Wisdom Rd.
  Dallas, TX 75236
  (972) 708-7371
  nrsi_ipub@sil.org