Keyman Developer 16 Tutorial

This tutorial is based on version 15. This tutorial is in the process of being updated for version 16.

Overview

Learning Objectives

1. Will create a Keyman project. (**Session 1**)
2. Will modify a Keyman project to add special characters using the code facility. (**Session 2**)
3. Will test a desktop keyboard. (**Session 3**)
4. Will create an icon for the keyboard. (**Session 4**)
5. Will create on-screen image of keyboard. (**Session 5**)
6. Will add a touch keyboard. (**Session 6**)
7. Will test a touch keyboard. (**Session 7**)
8. Will produce a keyboard package for distribution. (**Session 8**)
9. Will distribute a keyboard package. (**Session 9**)
10. Link(https://help.keyman.com/developer/16.0/guides/lexical-models/tutorial/))">Will create a lexical model for predictive text (Session 10 - on Keyman site)
11. Distribute lexical model to Keyman applications (Session 11 - on Keyman site)

In this tutorial we will create a keyboard for the Dagbani language.

Introduction

When we want to produce a keyboard, there is some planning we need to do first before we began creating our keyboard. We need to address the following questions

- Who is the keyboard for?
- What is the alphabet of the language?
- What is previous history for the input method for keying on the computer for this language.

The answers from these questions will inform how us how to the keyboard should be created.

If we have some experience designing typing keyboards for computers, the interface for designing touch keyboards looks deceptively similar. But the additional steps needed to make a touch layout work are not obvious. This tutorial walks we through the process of creating our first touch keyboard as well as a desktop keyboard as part of a complete keyboard solution in Keyman Developer. With Keyman we can have just one project to create a keyboard for a given language for all platforms.

We will need to have Keyman Developer installed on a Windows computer. Additionally, we may want keyman to be installed on one or more other devices on which we can test our keyboard.

This tutorial is based keyman Developer version 15.

Dagbani

In this tutorial we will create a keyboard for the Dagbani language of Ghana. The keyboard is for the Dagbani local community in Ghana to have a desktop and mobile keyboard for writing their language.

The alphabet for Dagbani is as follows:

a b ch d e ɛ f g gb ɣ h i j k kp l m n ny ŋ o ɔ p r s sh t u w y z ʒ '

Below are the special characters we will need to add to the Dagbani desktop keyboard. In the Key-in for desktop column, "; + e" means we press the semi-colon key then press the e key.

Name	Symbol	Unicode Value	Key-in for desktop
Open e	ɛ	U+025b	; + e
	Ɛ	U+0190	; + E
Open o	ɔ	U+0254	; + o
	Ɔ	U+0186	; + O
Eng.	ŋ	U+014b	; + n
	Ŋ	U+014A	; + N
Gamma	ɣ	U+0263	; + g
	Ɣ	U+0194	; + G
Ezh	ʒ	U+0292	; + z
	Ʒ	U+01b7	; + Z

We would use a touch keyboard for the mobile devices. We use the long-press to handle our special characters as below.

Name	Key	Long-press
Open e	e/E	ɛ/Ɛ
Open o	o/O	ɔ/Ɔ
Eng	n/N	ŋ/Ŋ
Gamma	g/G	ɣ/ Ɣ
Ezh	z.Z	ʒ/Ʒ

Here is a link to the pdf document that covers the special characters of the Dagbani language and how to produce them in the keyboard. We will need this pdf for the tutorial

Important differences about touch keyboards

In this tutorial, we distinguish between touch keyboards, which are pressed by fingers on a screen, and typing keyboards, which involve tapping real keyboard keys. Touch keyboard layouts must be treated differently than typing keyboards because they are primarily visual keyboards. Typing keyboards are usually learned by the feel of the keys and the use of muscle memory (where a repeated action becomes automatic), which is why sequences of keys are often used to produce a single character. In addition, for many typing keyboards, the user is expected to memorize a series of keystrokes to produce a desired character. For example, when laying out a typing keyboard, we might use the following sequence to produce the letter open o:

; + **o** = **ɔ** (that is, typing **;** followed by **o** produces **ɔ**).
Users must have a typing guide of some sort to remind them, but this combination is quickly learned. The combination might use a deadkey, where the semicolon (the deadkey) produces no visual output on the screen, but typing **o** produces **ɔ**. Or the keyboard might display the semicolon when it is typed, then replace it with **ɓ** when **b** is pressed. Typists will master this quickly and throw away the guide. Experienced typists won't need to look at the hardware keys when they type.

On a touch keyboard, however, there is an understanding that WYTIWYG (What you touch is what you get). That means that the letter **ɔ** ought to be visible somewhere on the keyboard. We'll look at our options for touch keyboards later in this document.

Plan beyond the touchscreen

We may think we only want to create a touch keyboard layout and don't want to bother setting up a typing keyboard layout. However, the distinction between a computer and a mobile device is becoming less and less. Often a tablet will be paired with a Bluetooth keyboard to enable the user to type faster. The minute this happens, our keyboard will not work as expected. Long press features don't work on the attached keyboard. That is why when we create a keyboard package in Keyman Developer, it is always assumed that we will produce both and bundle them together.