

Virtual Keyboard/On-Screen Keyboard

Limnor ships with an On-Screen Keyboard performer, which uses the on-screen keyboard comes with Windows. If you are not satisfied with the functionality of it, you may create your own on-screen keyboard using Limnor. This tutorial shows you how to do that. It is extremely easy and totally flexible.

[Make keys](#)

[Send more than one key strokes](#)

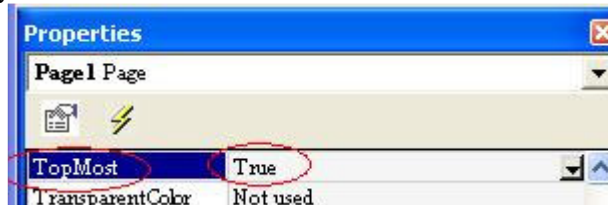
[Send special key strokes](#)

[Key Animation with Image List](#)

[Multi-language Virtual Keyboard](#)

Make keys

You may use Button performers to create keys for your on-screen keyboard. Suppose we create a page to be used as an on-screen keyboard, usually it is a good idea to set the TopMost property of the page to True, because usually we want an on-screen keyboard to be always on the front of all other windows:

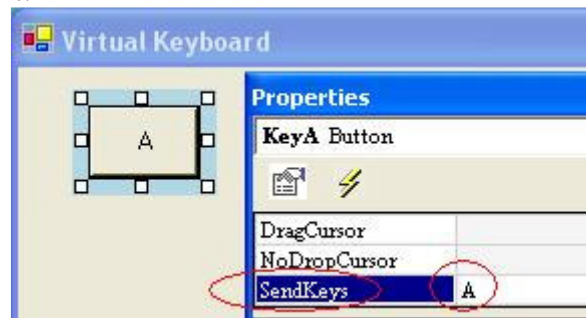


Let's create a Button performer on a page for the first key on our on-screen keyboard:



You may show images on the button when key-pressed and key-released to make it look cool. We will talk about it later.

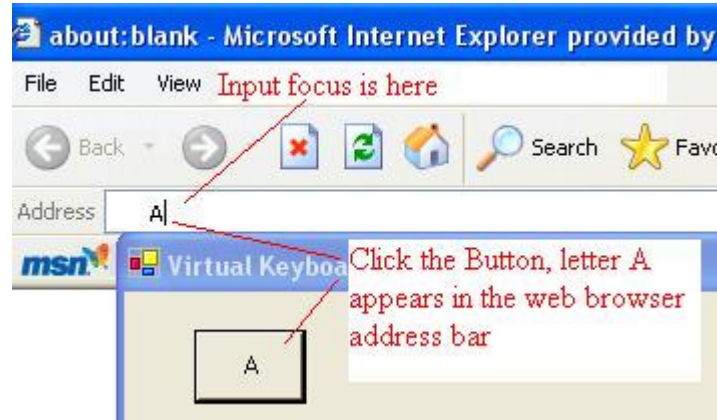
To make a normal Button performer to be a virtual key for an on-screen keyboard is extremely easy. You just set its "SendKeys" property to whatever key strokes you want this button to represent:



At runtime, when the user clicks the Button, the key strokes represented by the "SendKeys" property will be sent to the window having the input focus.

Virtual Keyboard

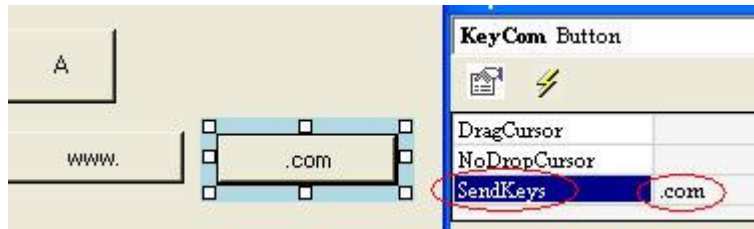
For example, if we open a web browser, and have the input focus inside the address bar, and clicks on our virtual key (the Button), the key stroke, in this example, “A”, will be sent to the address bar:



You see that by simply setting “SendKeys” property of a Button, you get a virtual key. That is why we say it is extremely simple.

Send more than one key strokes

Better than a standard keyboard, you can make a virtual key to send more than one key stroke. For example, if you make an on-screen keyboard to allow your users to input web addresses, you create one virtual key to send key strokes “www.”, and another virtual key for “.com”. You just need to set the “SendKeys” property to “www.” and “.com” respectively:



Send special key strokes

There are key codes for you to send special key strokes. For example, to simulate pressing Backspace, use “{BS}”:



We will give you a full list of special key codes below. The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses () have special meanings to “SendKeys” property. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use “{+}”. To specify brace characters, use “{{}}” and “{}}”. Brackets ([]) have no special meaning to “SendKeys” property, but you must enclose them in braces.

Virtual Keyboard

To specify characters that aren't displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the codes in the following table.

Key	Code
BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL or DELETE	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER} or ~
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS or INSERT	{INSERT} or {INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC} (reserved for future use)
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

Virtual Keyboard

Keypad add	{ADD}
Keypad subtract	{SUBTRACT}
Keypad multiply	{MULTIPLY}
Keypad divide	{DIVIDE}

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes.

Key Code

SHIFT +

CTRL ^

ALT %

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "+(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "+EC".

To specify repeating keys, use the form {key number}. You must put a space between key and number. For example, {LEFT 42} means press the LEFT ARROW key 42 times; {h 10} means press H 10 times.

Key Animation with Image List

Note that the skills shown in this section are not special for virtual keys. They are generic skills.

When you use your hand to type on a keyboard, the key on a keyboard first gets pushed down and then it pops back to its original position. You may use two images to visually simulate the push down and pop-up states.

A Button performer fires Mouse-Down and Mouse-Up events. At Mouse-Down event, we may let the button display an image representing push-down state; at Mouse-Up event, we may let the button display another image representing pops-up state (normal key state).

To display an image on the button, we may set its "Image" property to a file. So we may set "Image" property at Mouse-Down and Mouse-Up events. This will work. But loading images from files is slow. In this section, we show a method to use ImageList performer for a better performance.

An ImageList performer can hold a list of images in memory. Assigning an image from the ImageList performer to a Button will be much faster than loading the image from a file.

The Button performer has an "ImageList" property, which indicates which ImageList performer this button will get images from. It has another property "ImageIndex", which indicates which image from the ImageList the will be displayed on the button.

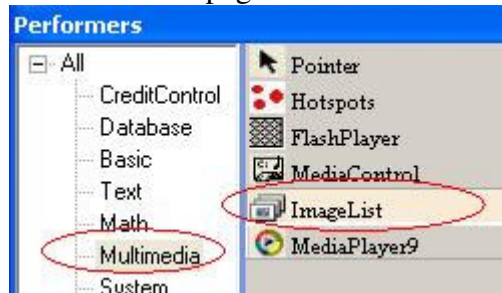
ImageIndex=0 indicates the first image, 1 indicates the second image, and so on.

Virtual Keyboard

At Mouse-Down and Mouse-Up events, we just need to switch ImageIndex, instead of re-loading image from file.

Step 1 – Create an ImageList performer


Let's add an ImageList performer to the page:

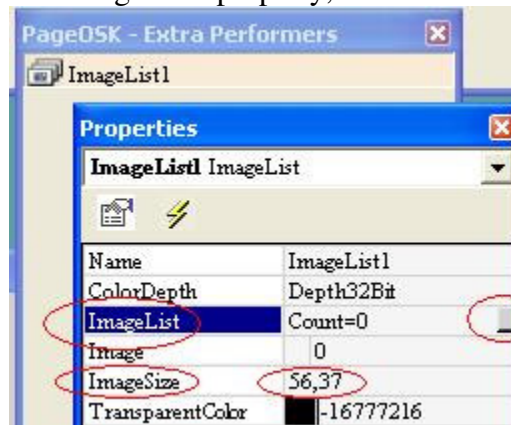


It appears in the Extra-performers window:

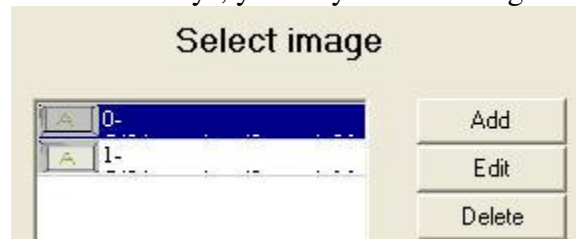


Step 2 – Set properties and add images

Right-click it and choose “Properties”. Set its ImageSize property to the Width and Height of the button. Select “ImageList” property, click  to add images:



For each button used as a virtual key, you may add two images, one for key-up state and one for key-down state. The following diagram shows two images for key “A”. If you design a virtual keyboard with 50 keys, you may use this image list to hold 100 images.

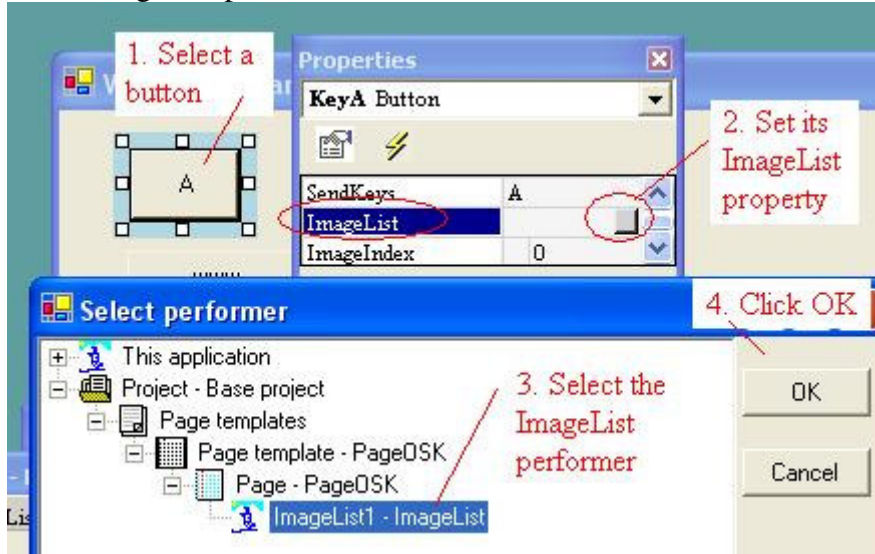


The key images used in this sample are very ugly. You certainly will make your own cool images.

Step 3 – Assign the ImageList performer to virtual keys

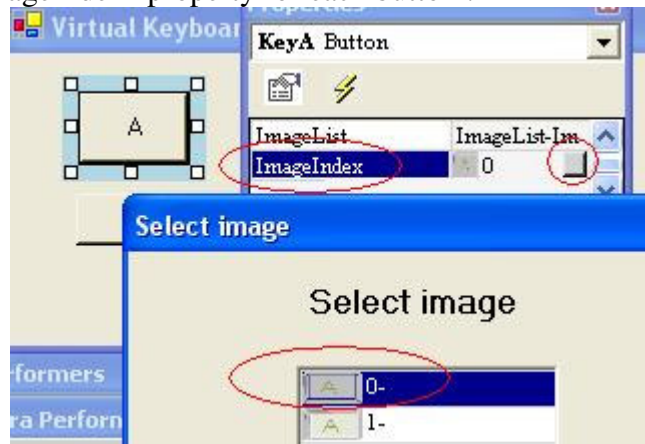
Virtual Keyboard

Each virtual key is a Button performer. Set the “ImageList” property of each Button performer to the ImageList performer created above.



Step 4 – Set Button Image

We need to show key-up image for all buttons which are used for virtual keys. You may do it by setting “ImageIndex” property for each button.

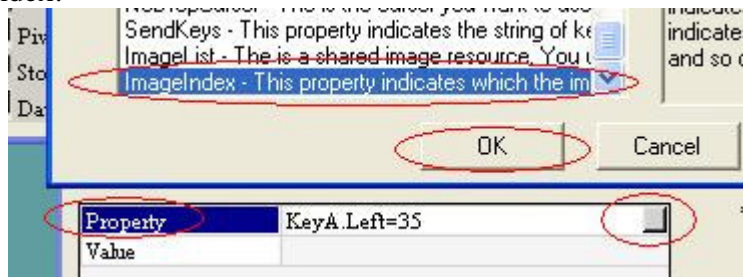


Step 5 – Set Text property to empty

Because we are using images on the button, we do not want button caption any more.

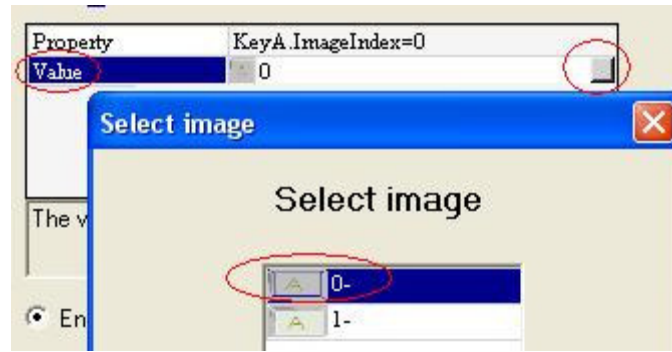
Step 6 – Create an action to show key-up image

Right-click on the button; choose “Make action”; choose “Set property”; give an action name, say, ShowKeyA_Up. The Action Data dialogue box appears. For “Property”, choose ImageIndex:



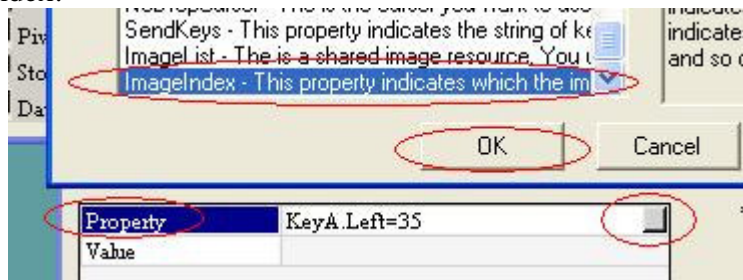
For “Value”, select the image you want to use for the key-up state of this button:

Virtual Keyboard

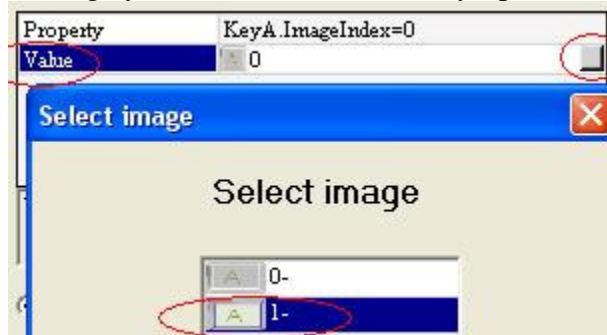


Step 7 – Create an action to show key-down image

Right-click on the button; choose “Make action”; choose “Set property”; give an action name, say, ShowKeyA_Down. The Action Data dialogue box appears. For “Property”, choose ImageIndex:

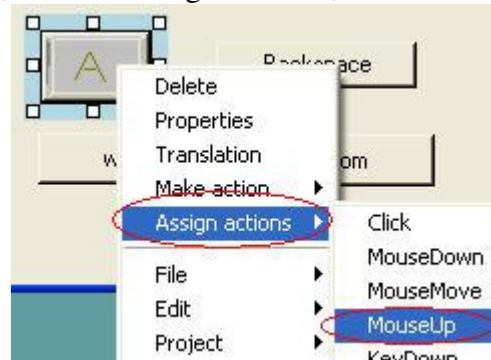


For “Value”, select the image you want to use for the key-up state of this button:



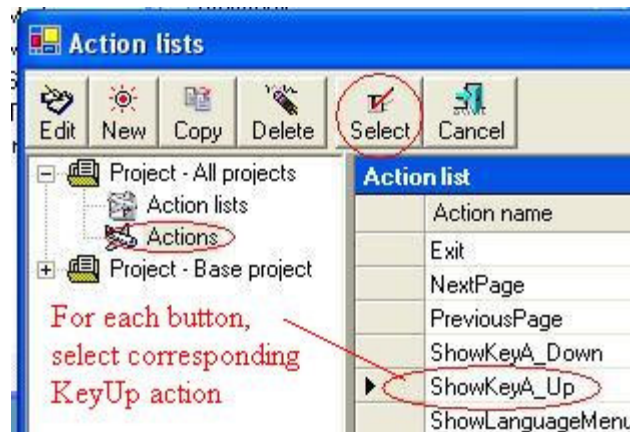
Step 8 – Assign the action made in Step 6 to the MouseUp event of the button.

Right-click on the button; choose “Assign actions”; choose “MouseUp”:

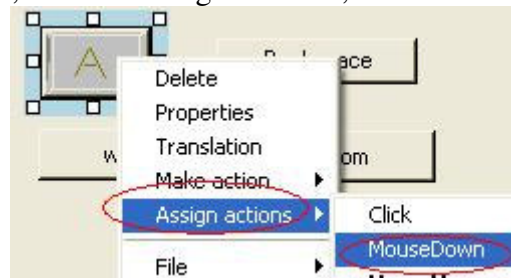


Select “Actions”; select the action made in Step 6; click “Select” button:

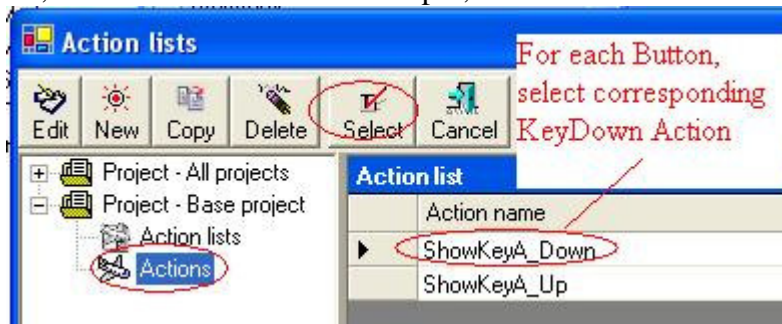
Virtual Keyboard



Step 9 – Assign the action made in Step 7 to the MouseDown event of the button.
Right-click on the button; choose “Assign actions”; choose “MouseDown”:



Select “Actions”; select the action made in Step 7; click “Select” button:



Repeat step 3 to step 9 for all the buttons used for virtual keys.

We are done making our own key animations.

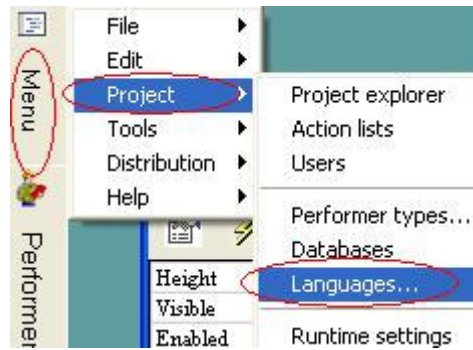
Multi-language Virtual Keyboard

Limnor has built-in multi-language support. That means the user may instantly switch languages and the program UI and contents will be automatically switched to the selected language. Using this feature, it will be easy to make a virtual key to send different key strokes for different languages.

The skills show in this section are not special to build on-screen keyboard. They are actually generic skills for developing multi-language applications.

Use “Languages...” menu under “Project” menu to determine which languages your program can support:

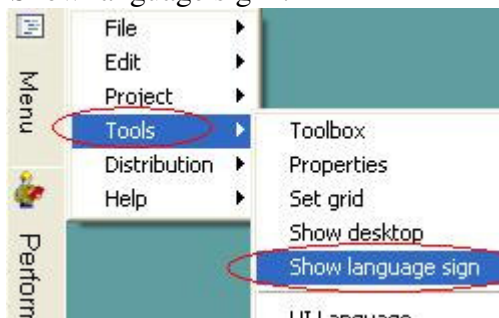
Virtual Keyboard



You may let your application to support as many languages as you want. The languages listed under “Languages used” are the languages supported by the application:



The way to develop multi-language applications in Limnor is very easy. You develop a multi-language application in the same way as you develop a single-language application. You just need to be aware of the language currently selected while you are doing development. To help you know the current language, you may turn on the language-indicator. Choose menu “Show language sign”:



A language-indicator appears:

Virtual Keyboard

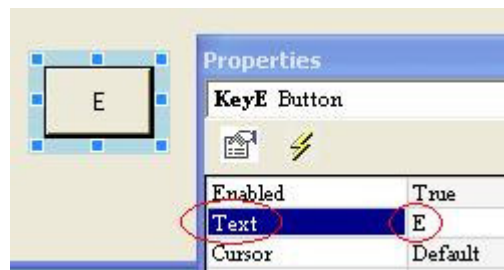


You may use mouse to drag the language-indicator and move it around the screen so that it will not block the screen.

You may right-click the language-indicator to show language selection menu to switch language:



Suppose now we select English as the current language. We create a button as a virtual key for letter "E":



To let it send key stroke "E", we set its "SendKeys" property to "E":

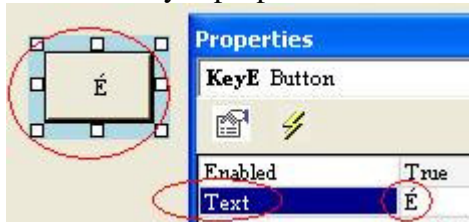


Now we switch language to French:



Now if you check the "Text" and "SendKeys" properties of the button, they are still "E". This is because we have not set French contents yet and Limnor automatically uses available contents.

Now let's change "Text" and "SendKeys" properties to "É" as an example:



Virtual Keyboard

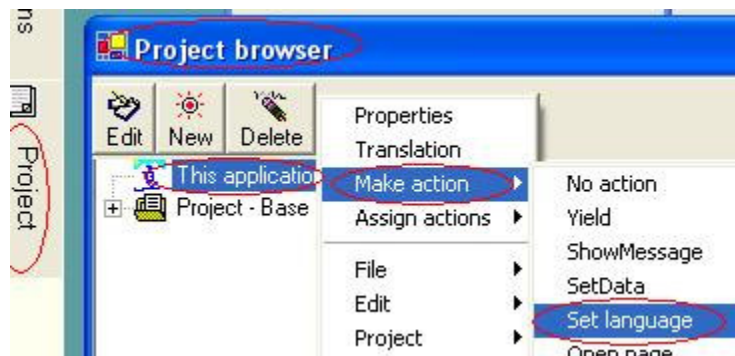


Note that Limnor remembers your setting of “Text” and “SendKeys” properties for all languages. You may switch back to English and you will see “Text” and “SendKeys” automatically changed to “E”. If you switch to French, they change to “É” again.

We need to provide a user interface to let users choose language at runtime.

The Application performer has a “Set language” method, which we may use to create actions to set language.

Open Project Browser, right-click “Application”, choose “Make action”, choose “Set language”:



Give an action name, say, **SetToEnglish**. Choose English as the action parameter:



We create another action named **SetToFrench**, and choose French as the action parameter:



In this example, we design such a user interface to let users select language: two pictures showing British and France flags respectively and two radio buttons beside each picture:



We create two actions for the radio buttons: **CheckEnglish** makes the radio button beside the British flag checked; **CheckFrench** makes the radio button beside the France flag checked.

We create two action lists: **SwitchToEnglish** includes actions **SetToEnglish** and **CheckEnglish**; **SwitchToFrench** includes actions **SetToFrench** and **CheckFrench**.

We assign action list **SwitchToEnglish** to the “Click” event of British flag and the “Checked” event of the radio button beside the British flag.

Virtual Keyboard

We assign action list **SwitchToFrench** to the “Click” event of France flag and the “Checked” event of the radio button beside the France flag.

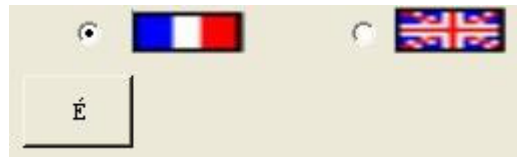
We also assign action list **SwitchToEnglish** to the “AfterShow” event of the page so that English is the language selected when the page is loaded.

Now we may press F2 to run our sample application. Click on the radio buttons or the flags; the language is instantly switched.

When English is selected, click button “E”, key stroke “E” is sent:



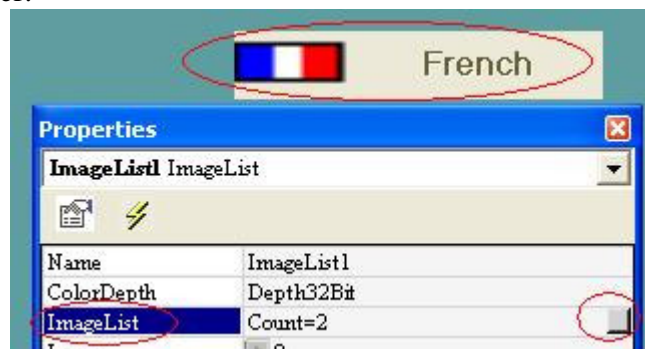
When French is selected, the button text automatically switches to “É”. Click the button, key stroke “É” is sent:



In this example, we use button text to show what the button will do. The button text automatically switches to the language selected. If you want, you may also change the font when the language changes. Just set the “Font” property for different languages, as we set Text property for different languages.

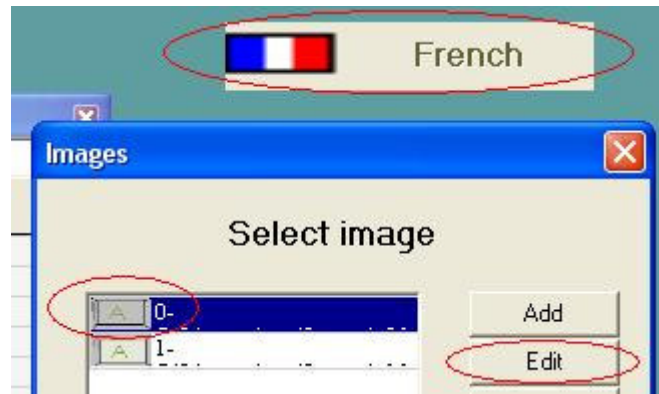
If we do not use button text, we use images on buttons, the images may also automatically change when language changes. If you use “Image” property to show image on a button, simply setting “Image” property for different languages. If you use “ImageList” and “ImageIndex” properties to show images on a button, you just need to assign images for different languages.

We first switch the language we want to load images. We set “ImageList” property of the ImageList performer:



We then use the Edit button to replace all the images in the image list. Note that we do not add new images, but just replace existing images. As setting other multi-language properties, when replacing existing data (images), the data for other languages are not erased.

Virtual Keyboard



As shown in the above diagram, the two images are assigned when English language is selected. Now, we select an image, click Edit button to select a French version image for this image. This will not erase the image set for English.

Now Press F2 to test it. Select different languages, you will see the button KeyA changes images automatically.

=== EOF ===

©2004 Longflow Enterprises Ltd. All rights reserved.