PDA Keyboard Simulator Design and Implementation

WANG Jia-ping (200228013202862)

Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100871, China

Abstract: This paper designs a simulator of PDA keyboard according to the specification given in the course. It presents a layered solution based on ActiveX and Dynamic-HTML technology and finally meets all requirements of the specification.

1. Requirements

A PDA, according to the given specification, have one screen that can display English (ASCII)/Chinese (Unicode-CJK) characters, and a keyboard with 40 keys. Now I will build software component to provide the ability of simple text editing. This part includes:

1. English characters inputting.
2. Chinese characters inputting.
3. Punctuation and special character inputting.
4. Simple editing such as moving cursor, deleting, page down/up.
5. Provides methods for switching inputting states among the above.

As a simulator, I also need build software component to simulate the screen and the keyboard. This part includes:

1. A text box that can display all ASCII and Unicode-CJK characters.
2. A keyboard with 40 keys with visual effect of striking.
3. An executable file to host all components that mentioned above.

2. Design

2.1 Basic architecture

The simulator is divide into two layers, show as figure 1. The upper layer component is for screen and kernel logic (Implement text editing). It is an ActiveX component built by C++ code. The lower layer is for the keyboard, which is implemented as a DHTML document. This document will be render by IE (Microsoft's Internet Explorer) and also hosts the upper layer, an ActiveX component. The main reason for implement keyboard as a DHTML is to reduce the development cost. It needs much less codes in DHTML/JavaScript to build a wonderful appearance of a keyboard with animation effects then in GUI/C++. For interacting with the keyboard built in DHTML/JavaScript, I design screen and kernel part as an ActiveX component.

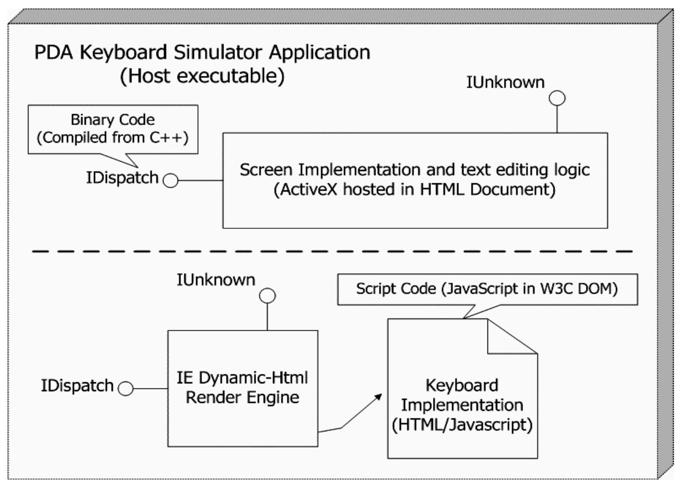
PDA Keyboard Simulator Application
(Host executable)
IUnknown
Binary Code (Compiled from C++)
IDispatch
Screen Implementation and text editing logic (ActiveX hosted in HTML Document)
IUnknown
Script Code (JavaScript in W3C DOM)
IDispatch
IE Dynamic-Html Render Engine
Keyboard Implementation (HTML/Javascript)

Figure 1. Components deployment

2.2 Keyboard and screen layout

The screen is divided into two panels: Text and Status. The upper panel is text area, which displays texts inputted and blinking cursor. The lower panel is status area that displays running states and candidate characters when inputting Chinese characters or punctuations. The Text area will automatically swap to new line when reach the end of line and rolls up when reach the bottom of the text area.

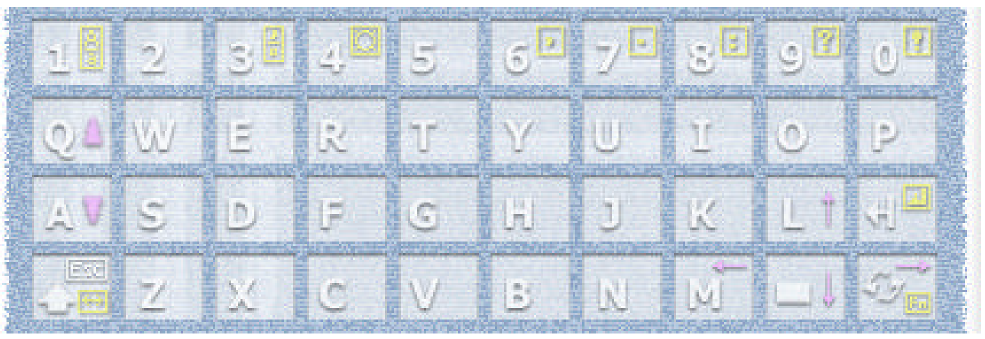

Figure 2. Keyboard layout

As show in figure 2, the keyboard composed of 40 buttons, 10 columns and 4 rows. Every button is of the same size marked with icon corresponding to its function. Some buttons is assigned with more then one function and will perform different functions in different states. Normally the function of the button is corresponding to the bigger white symbol marked on it. Yellow symbol works only when SHIFT key is latest pressed (the keyboard is shifted) and the pink symbol functions only in MOVING CURSOR state. Detailed functionality is stated in chapter 2.3.

2.3 Working state and switching

Kernel logic works like automata. It maintains current working state and switches from one into another according to incoming key striking. Working states can be one of the following:

1. English inputting (abbreviate to EI)
2. Chinese inputting (CI)
3. Moving cursor (MC)
4. Selecting candidate characters (CS)
   (includes Chinese characters, punctuation and special characters)
5. Selecting system functions (FS)

I divide the five states into two levels:

1. Level 1: EI and CI
2. Level 2: MC, CS and FS

Figure 3 demonstrates states switching logic. Initially it gets in one of the level1 states (EI or CI, EI is the default one according to specification), then jump up to one of level2 states for some reason (global hotkey pressed T3, T4, T2 or selecting candidate characters T2). When jumping up to level2 state, all information about original state (a level1 state) is preserved for restoring. A level2 state may not explicitly switch into level1 state, it returns instead. When a level2 state is finished (canceled or done), the current state information will be discarded and the original state (a level1 state)

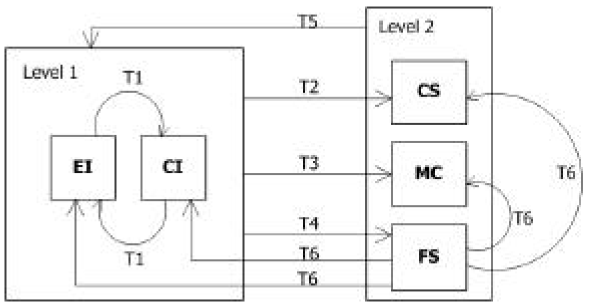

Figure 3. State diagram

will be restored as current state. The route T5 in figure 3 represents for returning form level2 to level1. Certainly level2 state is allowed to explicitly switch to another state. The state FS (Selecting System functions) may switch to CS, MC, EI and CI (T6) according to user's selection.

When user clicks the keyboard, kernel will filter all incoming keys and change current working state if necessary. Do not take for granted that one click may lead to at most one step of state changing. Take CI for example; when there is some part of PinYing code was typed and the match candidate Chinese character is display, the current state is level2 CS. When a new click comes in (such as 'n'), (in CS state, 0-9 is legal selection and any other input will lead CS to being canceled) CS is canceled and returns to CI. Then the incoming key 'n', instead of being discarded, is turn into new click input again, then CI receive it and update the PinYing code to generate new candidate Chinese characters list. Finally the state was switched into CS again, thus the 'n' click lead to two steps of state changing.

3. Implementation

3.1 Overview

I use Visual C++.Net to develop core components: KeySim.exe (Host executable) and SimCore.dll (ActiveX) and use Dreamweaver and Photoshop to product HTMLs and images associated. PinYing-Chinese mapping and Chinese Phrase data is hard-coded in source code as static const Unicode-string arrays. I use binary-search when locating Chinese candidate character entries and candidate associated phrases. I use two customized Edit Box common controls to implement the PDA screen, one for Text area and another for Status area. I disable all interactive functionalities of Edit Box for making it looks like a screen. SimCore.dll, the ActiveX, exposes two methods: SendKey (to be invoked when some key is pressed) and Print (called when application starts up to display the welcome text).

3.2 source code

3.2.1 C++ code

Open KeySim.sln in VC.Net (I hope you have the MS VS.Net installed ^_^, because this workspace file is NOT compatible with VC 6.0), you can find there are two VC projects, one for KeySim.exe and another for SimCore.dll. I don't think it makes sense for writing comments function by function, I will summary several key classes and functions with comments. I commented some of functions and specified every argument of function with IDL (Interface definition language).

◆ **In Project KeySim.Exe:**

CKeySimApp (KeySim.h KeySim.cpp): Derived from CWinApp provides member functions for initializing host application (and each instance of it) and for running the application. In CKeySimApp::InitInstance, I perform COM register for SimCore.dll and create an instance of CkeySimDlg for loading and rendering the HTML document.

CKeySimDlg (KeySimDlg.h KeySimDlg.cpp): Derived from CDHtmlDialog (available only in MFC 7.0). It creates a dialog box that use HTML rather than dialog resources to implement their user interface. In CKeySimDlg::OnInitDialog, I navigate to Htmls\main.htm after normal initialization.

◆ **In Project SimCore.DLL**

CKernelApp (Kernel.h Kernel.cpp): Derived from COleControlModule provides member functions for initializing OLE control module.

CImeMapping<int> (ImeRoot.h): A class template provides management for PinYing-Chinese mapping and method for locating a PinYing entry. CImeMapping::Search returns all matched PinYing-Chinese mapping entry Ids according to the given PinYing code.

CPDA_SimCtrl (PDA_SimCtrl.h PDA_SimCtrl.cpp): the kernel class of the ActiveX control, derived from COleControl, hosts the two panels of CScreen as Text area and Status area, handles key striking event from HTML, switches working states and implement all functionalities of simple text edit. CPDA_SimCtrl::HandleInXXXX, per-state event handler, filters key strike events in specific working state, maintain private data of current state. In CPDA_SimCtrl::OnCreate, initialization when ActiveX component created, I create two instances of CScreen, one for Text area and the other for Status area. CPDA_SimCtrl::SendKey is main OLE automation method called by JavaScript in the HTML document, it is the first pipe that filters all key striking events, match global hotkeys and routing key striking events to pre-state event handler (CPDA_SimCtrl::HandleInXXXX). CPDA_SimCtrl::ShiftUp change the state from level1 to level2 as T2,T3,T4 in figure 3; CPDA_SimCtrl::ShiftDown return the state from level2 to level1 as T5 and CPDA_SimCtrl::SwitchTo change state from one level1 state to another level1 state as T1.

CScreen (screen.h screen.cpp): A customized Edit Box common control, derived form CEdit. For making it looks like a screen panel, I disabled all interactive functionalities of the original CEdit by the means of overriding. All CScreen::OnXXXXXXX is overridden functions, in which I prevent the parent class (CEdit) from being notified of user interactive events. It also provides methods for Inserting, Deleting and Moving cursor.

3.2.2 JavaScript and HTML

All JavaScript and HTML code is in the file Htmls\Main.htm. You can open the file with any HTML editor, or even Notepad is ok. JavaScript code, within HEAD section, mainly handles the events fired in the HTML document and interprets those user actions (events) as keyboard striking then drive the ActiveX (SimCore.dll) to run by invoking the SendKey method of SimCore.dll. In additional, a feature is also implemented here: after user holds a key down for a certain period (600ms), the keyboard will automatically generate key strikes of the button 12.5 times per second until the key is released.