实 验 十四 创建软键盘

14.1 实验目的

窗体资源与状态是窗体程序要操作的重要内容，本实验实现一个软键盘程序可向其它窗体程序输入字符，它综合运用窗体和控件资源管理和属性，通过重载窗体类改变窗体的成员，结合事件机制产生自定交键盘消息。

14.2 Windows 窗体

窗体资源是窗体程序的编程元素，要能提供用户灵活的界面显示，就需要掌握窗体之间的关系包括尺寸大小、移动位置等改变窗体属性的方法。

Windows 系统管理系统资源的方式多数通过句柄对象，窗体句柄对窗体的标识，系统有大量通过窗体句柄操作窗体的方法接口，例如内核函数 GetDesktopWindow 可获取桌面窗体句柄。

14.2.1 桌面窗体

系统启动时创建桌面窗体，它绘制整个屏幕是其它窗体体对象的基础。墙纸是在系统的注册表中定义的一个位图文件 (.bmp)，用来绘制桌面背景。应用程序可使用 SystemParametersInfo 函数修改或设置墙纸参数。

14.2.2 应用程序窗体

窗体应用程序最少创建一个窗体对象又称为主窗体，它是用户和应用程序交互接口界面。应用程序也会创建其它窗体，这些窗体都用来绘制结果或从用户接收输入。窗体程序运行时一个任务栏按钮与其关联，按钮的按下表示窗体处于激活状态。

窗体应用程序具有一些组成元素，包括标题条、菜单条、窗体菜单、最小化按钮、最大化按钮、关闭按钮；可拉伸的边界，客户区，水平滚动条，垂直滚动条。图14-1表示了这些元素的分布。

图14-1中的客户区是窗体用来显示输出的区域，比如文本与图形都是输出在客户区。在图14-1中标题条，菜单条，最小化最大化按钮，可拉伸边界，滚动条合在一起全部都称为非客户区。系统管理非客户区内容，而用户程序管理客户区的显示与行为。应用程序在创建窗体时将在标题栏区显示一个图标和一行文本，标题栏使用户可以拖动窗体。

应用程序创建窗体时需要设置类名称与标题信息，window class 名称，程序须注册窗体类，类定义了窗体的大部分外观与行为特性，最主要的部分是窗体过程 (window procedure)，用来接收和处理所有被发送到窗体的输入消息。window Name 的显示如果窗体有标题栏的则在标题

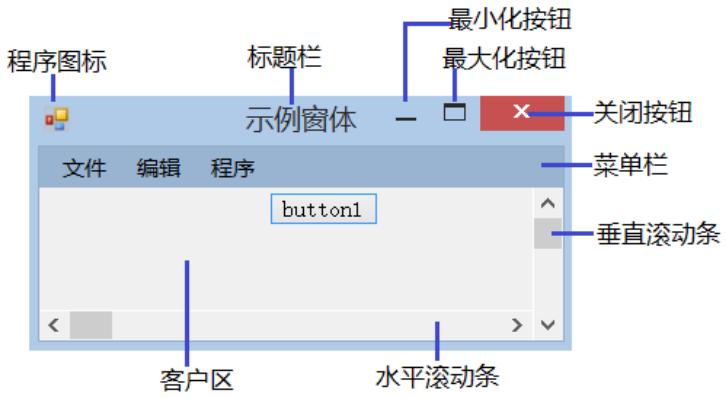

图 14-1 窗体区域

栏显示，控件则根据其特点显示在其显示区域，如按钮直接显示至主要区域；一些控件则不显示，如 ListBox 控件和流动条等。SetWindowText 函数设置窗体名称，GetWindowTex 获取窗体标题字符串。

14.2.3 窗体资源与关系管理

Windows 系统对资源管理的典型方法是采用句柄，句柄是系统用唯一的整数代表某个资源。大多数窗体函数都需要窗体句柄作为参数，SetActiveWindow 方法通过指定的窗体句柄将其激活。

Windows 系统只有一个当前激活的窗体，系统把用户的输入消息发送到激活的窗体。多个窗体的重叠绘制表示窗体的前后关系，显示的 Z-Order 关系。激活状态的窗体与前端窗体并不等同，

窗体的所有关系。SetForegroundWindow 窗体显示状态有：激活窗体，失效窗体。窗体的可视性属性有：最大化，最小化，恢复原状。

应用程序向系统申请创建及通知结束，窗体资源的具体分配与销毁是由系统来完成的。SetWindowLong 用来设置窗体资源的关联数据，SetWindowLong, SetWindowPos 使新设置的窗体数据生效。

14.3 软键盘窗体程序设计

14.3.1 软键盘程序功能介绍

软键盘程序采用鼠标替代键盘向其它窗体程序输入字符，需要具有窗体资源的目标进程，它运行时具有以下特点：

1. 点击键盘程序不会改变处于激活状态的目标窗体。
2. 用户点击一个按钮，向当前激活程序发送键值，例如"A" 键。
3. 支持用户使用鼠标左键占击键盘窗体标题区后拖动。
4. 支持功能键例如"win" 键，"Delete" 键。
5. 支持组合键，例如"Ctrl+C"。

需要先运行记事本程序或者其它文本编辑软件作为当前激活程序，然后再运行本程序。图14-2是程序运行效果图，它向激活窗体输入键盘字符。

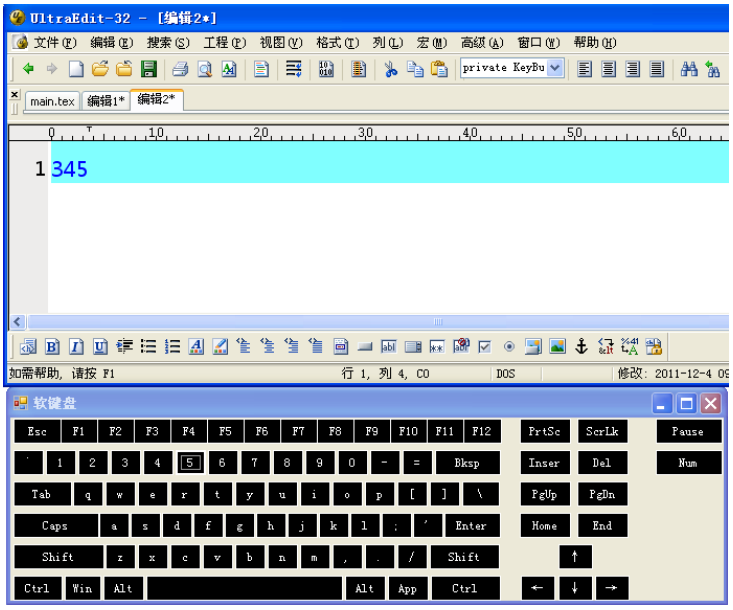

图 14-2 软键盘运行界面

14.3.2 原理说明

软键盘程序属窗体程序确与普通窗体程序有较大差别。软键盘窗体界面修改包括：窗体状态为非激活窗体，按钮外观及每个按钮增加键值属性。软键盘窗体消息处理重载：鼠标按下获取目标窗体句顶，向目标窗体发送模拟键值，鼠标移动重新激活目标窗体。

本程序有两个核心内容，一是软键盘窗体状态为非激活窗体，点击软键盘窗体中的按钮向其它窗体发送键盘字符；二是，本程序将由 Form 派生新类实现要求的窗体特性；二是从按钮按件派生新类，新类可设定新的显示样式，并能标识当前按键代表的键值。

本程序中 NoActivateWindow 类继承自 System.Windows.Forms.Form 类，继承时重载窗体的 CreateParams 参数，添加 WS_EX_NOACTIVATE 样式，这样窗体将不会得到焦点；窗体拖动是对窗体的特定鼠标消息进行了重载实现。主窗体类继承自 NoActivateWindow 类，则自动具有 NoActivateWindow 类特性。

键盘按键使用的是按钮控件，首先定义 KeyButton 类，它继承自 System.Windows.Forms.Button 类，在构造函数中重载了其显示样式，比如前景色与后景色。KeyButton 类还增加了一些新的功能，比如当按下 Shift 键时，其它按钮将显示另外一个字符；普通键盘字符按下将向目标激活窗体发送其所代表的字符。

14.3.3 项目关键代码

本实验代码较长，为减化代码排版，直接提供必要的源文件，用户可将文件添加到自己的项目中。表14-1列出了项目文件的组成与功能说明。新建一个窗体应用程序，路径为 d:\xue，项目名称为 softKB。其中主窗体文件是初始创建的窗体程序，用户须将其它源文件添加到

表 14-1 项目文件说明

文件名	功能说明
UnsafeNativeMethods.cs	封装了程序需要用到的内核函数。
NoActivateWindow.cs	派生自 Form 类，重载了窗体的创建状态，和对鼠标拖动的消息响应。
KeysMapping.xml	记录键盘虚拟码与字符对应关系。
NativeMethods.cs	封装鼠标和键盘消息结构。
KeyboardInput.cs	向目标程序发送键盘按键值。
KeyButton.cs	对 Button 类重新其外观样式，添加属性值，例如 KeyCode 属性用于标识当前按钮代表的键盘虚拟码。
FrmMain.cs	主窗体程序，由 NoActivateWindow 窗体类派生。

表 14-2 程序使用的 Windows 内核函数

函数名	功能说明
GetForegroundWindow	获取 windows 当前激活窗体的句柄值。
SetForegroundWindow	指定某窗体对象为 windows 的当前激活窗体。
GetKeyState	获取虚拟键的按下或抬起状态。
SendInput	向目标程序发送键盘按键值。

自己的项目中，注意修改源文件中的命名空间与项目的命名空间一致。项目中需要使用的 Windows 内核函数如表14-2，这些内核函数以类 UnsafeNativeMethods 名称被封装在文件 UnsafeNativeMethods.cs 中。

14.3.4 按钮控件的继承与使用

向窗体添加一个普通的按钮控件，使用解决方案资源管理器打开 FrmMain.Designer.cs 文件，找到 button1 控件名，将其类型由 System.Windows.Forms.Button 修改为 KeyButton，则普通按钮将变成被重载后的按钮控件。新的按钮控件将具有增加的属性，如 KeyCode 属性代表按键虚拟码。

14.3.5 窗体类的继承与使用

在新建的窗体代码中，将原来的派生类由 Form 改成 NoActivateWindow。向窗体添加如图14-2排列的 KeyButton 按钮，每个按钮需要指定其 KeyCode 值。表14-3列出了按键名称及其 KeyCode 值。

在类的构造函数前添加下面的代码：

```
IEnumerable<KeyButton> keyButtonList = null;
IEnumerable<KeyButton> KeyButtonList
{
  get
  {
    if (keyButtonList == null)
    {
```

表 14-3 键名称与 KeyCode

键名称	值	键名称	值	键名称	值
keyButtonEscape	27	keyButtonPrintScreen	44	keyButtonScrollLock	145
keyButtonCapsLock	20	keyButtonRShift	16	keyButtonQuestion	191
keyButtonPeriod	190	keyButtonComma	186	keyButtonNumLock	144
keyButtonTab	9	keyButtonReturn	13	keyButtonSemicolon	186
keyButtonMinus	189	keyButtonPlus	187	keyButtonBack	8
keyButtonOemPipe	220	keyButtonCloseBrackets	221	keyButtonOpenBrackets	219
keyButtonInsert	45	keyButtonDelete	46	keyButtonPageUp	33
keyButtonPageDown	34	keyButtonHome	36	keyButtonEnd	35
keyButtonTilde	192	keyButtonDown	40	keyButtonLeft	37
keyButtonUp	38	keyButtonRight	39	keyButtonLShift	16
keyButtonProcess	222	keyButtonWin	91	keyButtonSpace	32
keyButtonPause	19	keyButtonRControl	17	keyButtonLControl	17
keyButtonApps	93	keyButtonLAlt	18	keyButtonRAlt	18
keyButtonF1	112	keyButtonD1	49	keyButtonA	65
keyButtonF2	113	keyButtonD2	50	keyButtonB	66
keyButtonF3	114	keyButtonD3	51	keyButtonC	67
keyButtonF4	115	keyButtonD4	52	keyButtonD	68
keyButtonF5	116	keyButtonD5	53	keyButtonE	69
keyButtonF6	117	keyButtonD6	54	keyButtonF	70
keyButtonF7	118	keyButtonD7	55	keyButtonG	71
keyButtonF8	119	keyButtonD8	56	keyButtonH	72
keyButtonF9	120	keyButtonD9	57	keyButtonI	73
keyButtonF10	121	keyButtonD0	48	keyButtonJ	74
keyButtonF11	122	keyButtonK	75	keyButtonL	76
keyButtonF12	123	keyButtonM	77	keyButtonN	78
keyButtonO	79	keyButtonP	80	keyButtonQ	81
keyButtonR	82	keyButtonS	83	keyButtonT	84
keyButtonU	85	keyButtonV	86	keyButtonW	87
keyButtonX	88	keyButtonY	89	keyButtonZ	90

```
            keyButtonList = this.Controls.OfType<KeyButton>();
        }
        return keyButtonList;
    }
}
List<int> pressedModifierKeyCodes = null;
/// <summary>
/// The pressed modifier keys.
/// </summary>
List<int> PressedModifierKeyCodes
{
    get
    {
        if (pressedModifierKeyCodes == null)
        {
            pressedModifierKeyCodes = new List<int>();
        }
        return pressedModifierKeyCodes;
    }
}
```

在窗体的 Load 事件中添加代码，添加所有按钮的点击响应事件。

```
 try
{
    InitializeKeyButtons();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
// Register the key button click event.
foreach (KeyButton btn in this.KeyButtonList)
{
    btn.Click += new EventHandler(KeyButton_Click);
}
this.Activate();
```

余下的实现函数可参考提供的源程序。

14.4 作业

1. 根据文档中程序界面，设计完成软键盘程序。