基于PowerBuilder的窗口控件大小调整的实现

赵学锋　张金隆　蔡淑琴　金　鹏
(武汉华中科技大学管理学院　430074)

摘要：本文介绍了在PB中如何利用静态文本条来实现窗口面板的控件调整大小的基本原理，并通过窗口函数和用户事件结合实例给出了实现方法。

关键词：PowerBuilder　窗口　控件　调整

1 引言

大型数据库前端开发工具PowerBuilder(以下简称PB)，是目前非常流行的一种开发工具，它以其高效率、可视化、支持面向对象技术、标准的Windows界面等卓越性能为广大数据库应用软件开发者所青睐。

PB提供了树视图、列表框、下拉列表框、datawindow等很多控件，利用这些控件能开发出友好的窗口界面，但在实际使用中还是会碰到一些小问题。比如图1所示，左边的树视图和右边的列表框排列在一起时，因为显示的原因需要动态调整树视图和列表框水平方向的大小，即把树视图宽度减小的同时把列表框的宽度增大，但保持二者的相对位置不变，这也是很多软件所采用的一种窗口界面动态调整的技术。但PB所提供的控件没有自动改变水平方向大小的属性，需要通过编程来实现。本文通过分析在窗口面板中动态进行控件调整的基本原理，并结合实例给出了一个通用的实现方法。

2 窗口中控件调整的基本原理

在窗口面板中调整控件的相对大小的基本思想是通过一个静态文本条来实现的，其基本原理如下：在窗口的两个控件中间放置一个静态文本条，通过鼠标拖动静态文本条到一个新位置，然后根据文本条的所在位置增加或减少两个控件水平方向的宽度，达到调整控件的位置和相对大小的目的。

3 实现方法

下面通过一个简单实例来阐述如何通过拖动鼠标达到在窗口面板中自动调整控件大小的目的。图2是一个简单的实验窗口，左边是一个树视图控件(控件名为tv-1)，右边是一个数据窗口控件(控件名为dw-1)，在两个控件中间放置一个静态文本控件(控件名为st-vettical为了显示的方便，将它的背景色置为黑色)。以下是窗口面板控件大小自动调整的实现过程：

3.1 在窗口中定义实例变量

```
long il-hiddencolor=0 //使文本条隐藏的颜色，使之和窗口的背景色相适应
integer ii-barthickness=8 // 定义文本条的宽度
integer ii-windowborder=25 // 控件离开窗口左右两边的距离
dragobject idrg-Vertical [2] //定义拖动对象数组，存放控件名
integer li-maxheight // 窗口中控件的最大高度
```

定义il-hiddencolor是为了使拖动控件时能显示文本条，停止拖动时隐藏文本条；为了保证文本条在拖动时能可见，定义了文本条的宽度ii-barthickness。为了窗口美观和保证控件不致超出窗口之外，定义了一个控件离窗口左右边缘的最小距离ii-windowborder。为了程序的通用性，定义了拖动对象数组idrg-Vertical []，用来存放控件。

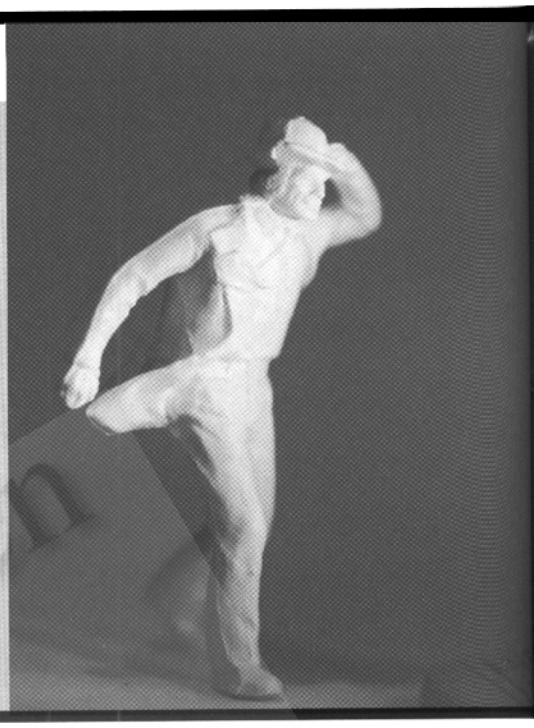
3.2 定义窗口函数

定义三个窗口函数：wf-resizebars、wf-refreshbars、wf-resizewindows，其中wf-refreshbars的主要作用是重画文本条，使得文本条能在下次拖动时能显示。wf-resizebars的主要作用是根据鼠标停留的位置改变文本条的位置和大小。Wf-resizewindows的主要作用是根据文本条的位置调整控件的位置和大小。

(1) 窗口函数wf-refreshbars

```
st-vertical.SetPosition(ToTop!) // 让文本条st-vertical始终处在其他控件的上面
```

Implementation of Size Adjustment of Windows Controlbox based on Powerbuilder



return 1 //返回值

(2) 窗口函数 wf-resizebars

st-vertical.Move (st-vertical.x, ii-WindowBorder) //将文本条 st-vertical 移动到新位置

st-vertical.Resize(ii-barthickness, li-maxheight) //调整文本条的大小

wf-RefreshBars() //调用窗口函数 wf-refreshbars

return 1 //返回值

(3) 窗口函数 wf-resizewindows

ll-Width = This.WorkSpaceWidth() //获取指定窗口工作区的宽度

//将控件 idrg-Vertical [1] 移动到新位置，保留控件离窗口左边和上边的距离

idrg-Vertical [1].Move (ii-WindowBorder, ii-WindowBorder)

//根据文本条的位置调整控件idrg-Vertical [1] 的大小

idrg-Vertical [1].Resize (st-vertical.x - idrg-Vertical [1].x, idrg-Vertical [1].height)

//将控件 idrg-Vertical [2] 移动到新位置，保留控件离窗口右边缘和上边缘的距离

idrg-Vertical [2].Move (st-vertical.x + ii-BarThickness, ii-WindowBorder)

//根据文本条的位置调整控件idrg-Vertical [2] 的大小

idrg-Vertical [2].Resize (ll-Width - idrg-Vertical [2].x - ii-WindowBorder, idrg-Vertical [2].height)

Return 1 //返回值

3.3 初始化文本条，为控件数组赋值

为了使窗口打开时，使静态文本条不可见并且为文本条指定大小，必须对静态文本条进行初始化。

idrg-Vertical [1] = tv-1 //为拖动对象数组赋值

idrg-Vertical [2] = dw-1

il-HiddenColor = This.BackColor //使静态文本条变为不可见

st-Vertical.BackColor = il-HiddenColor

if idrg-vertical [1].height>=idrg-vertical [2].height then //找出拖动对象最大的高度

li-maxheight=idrg-vertical [1].height

else

li-maxheight=idrg-vertical [2].height

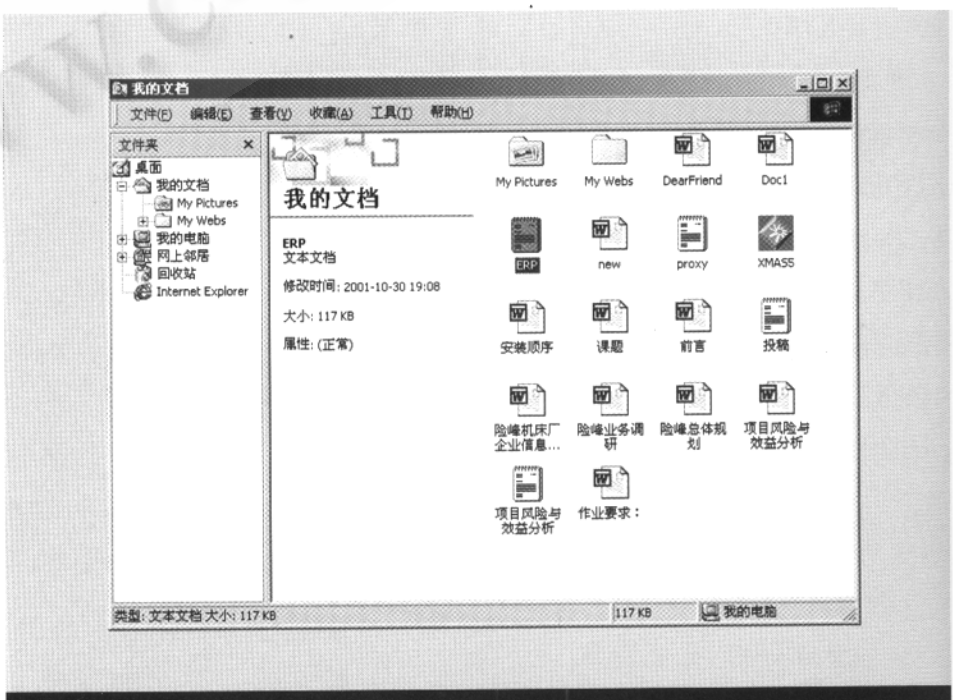


图 1 窗口面板中控件的自动调整

end if

wf-ResizeBars() // 调用窗口函数改变文本条的位置和大小

wf-Resizewindows() // 调用窗口函数改变控件的位置和大小

3.4 定义用户自定义事件，对静态文本条编程

使用PB开发应用系统的核心使围绕触发对象执行相应对象的事件代码。PB本身提供了很多事件，但在实际应用开发中辉由于特殊需要或为方便地实现某个功能而希望使用自己定义的事件，为了使用户在拖动静态文本条来实现控件大小和位置的调整，必须使用用户自定义事件。用户拖动文本条，要顺序执行三个动作：按下鼠标左键、鼠标移动拖动文本条、松开鼠标左键停止拖动，由此定义了三个用户自定义事件：

(1) 鼠标移动的用户自定义事件：mouse move

用户自定义事件名为mousemove，event-id为pbm-mousemove，其脚本如下：

// 当鼠标移到文本条上时，将鼠标指针变为横向箭头

If abs (Parent.PointerX() - this.x)<=ii-barthickness then setpointer (SizeWE!)

If KeyDown (keyLeftButton!) Then // 按下鼠标左键，将鼠标所在位置的x坐标赋值

This.x = Parent.PointerX() // 给文本条的x坐标

End if

(2) 松开鼠标左键，鼠标键弹起，表示拖动停止的用户自定义事件：mouseup

用户自定义事件名为mouseup，event-id为pbm-lbuttonup，当鼠标左键向上弹起表示停止拖动时，调用窗口函数wf-Resizewindows()，调整控件的大小和位置。其脚本如下：

This.BackColor = il-hiddencolor // 隐藏文本条

wf-Resizewindows() // 调用窗口函数，调整控件大小和位置

(3) 按下鼠标左键的用户自定义事件：mousedown 用户自定义事件名为mousedown，event-id为pbm-lbuttondown，当按下鼠标左键表示拖动即将开始时将静态文本条的背景色变为黑色，其脚本如下：

this.BackColor = 0 // 使文本条的颜色变为黑色，显示拖动轨迹

4 小结

本文介绍了在PB中通过鼠标拖动静态文本条实现窗口面板控件自动调整的基本原理，结合实例通过窗口函数和用户自定义事件给出了一种通用的实现方法及过程。上述实例只是从实现功能上来说明的，因此只使用了树视图和数据窗口两类控件。如果要实现其他类型的控件大小调整，只要对程序做一点调整就可以了，即在窗口的初始化程序中，将拖动对象数组重新赋值idrg-vertical [] 就可以了。如果要实现垂直方向的控件大小调整，可以定义一个静态水平文本条，仿照本文所提供的方法就可以实现。

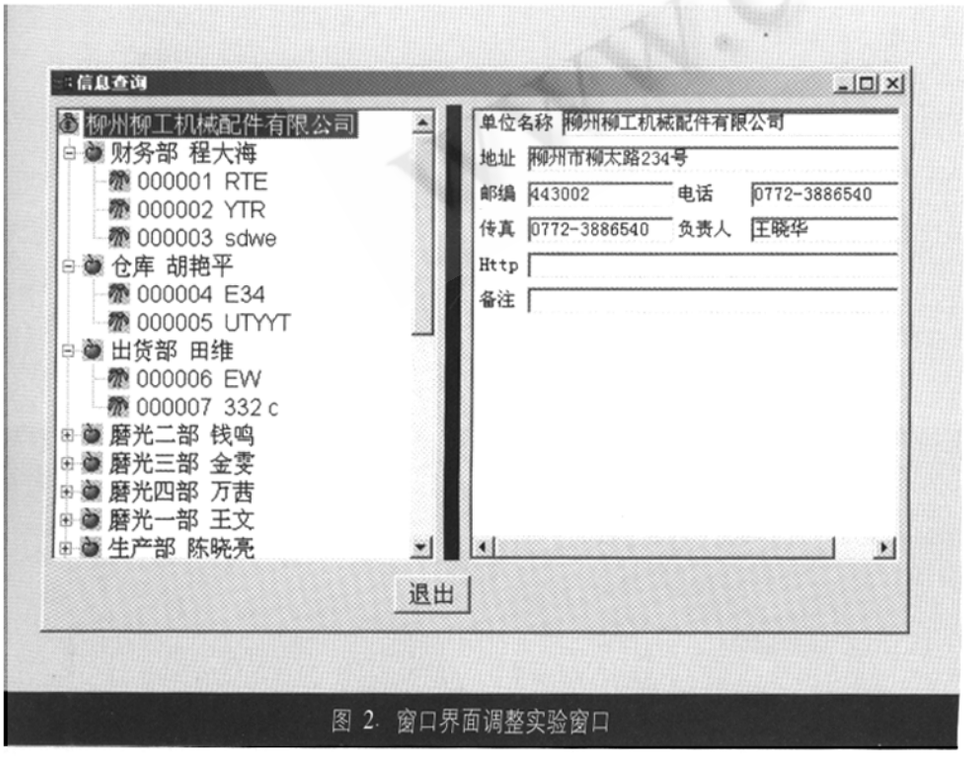


图 2. 窗口界面调整实验窗口

参考文献

1 吴洁明，PowerBuilder 6.0应用与开发 [M]，清华大学出版社，1998。

2 沃高工作室，PowerBuilder 6.0 应用开发指南 [M]，北京人民出版社，1998。

3 刘红岩，PowerBuilder 原理与应用指南 [M]，电子工业出版社，1999。

4 何田等，PowerBuilder 6.0程序员参考手册 [M]，水利水电出版社，1999。