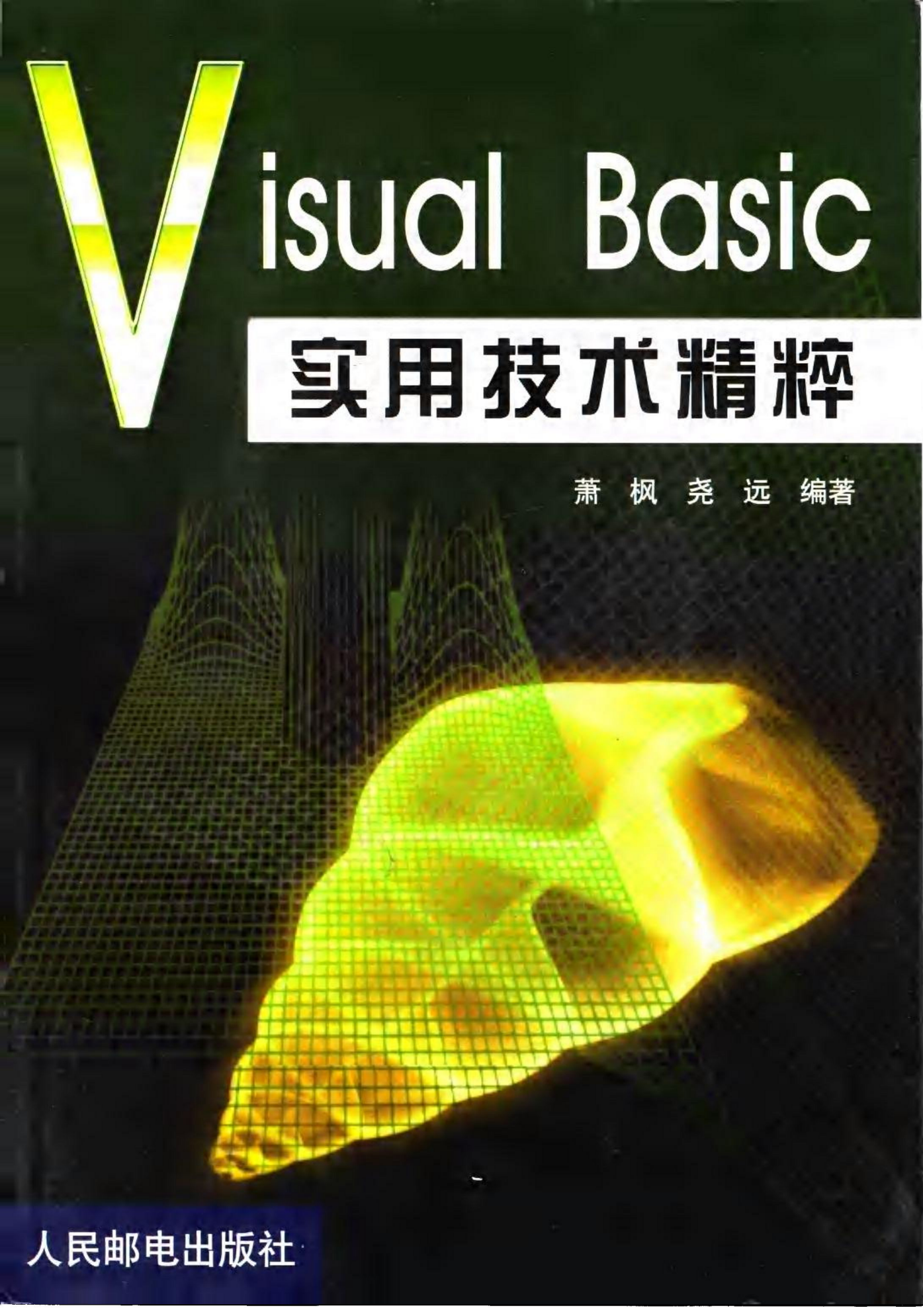
Visual Basic

实用技术精粹

萧 枫 尧 远 编著



人民邮电出版社

Visual Basic 实用技术精粹

萧枫 尧远 编著

人民邮电出版社

内 容 提 要

Visual Basic 是一套完全独立的 Windows 开发系统，它完全可以和 C 语言程序员使用的专业开发工具 Visual C++相媲美。本书将向读者展示如何完成一些读者可能认为只有 Visual C++才能做到的事情。书中展示了各种 Visual Basic 高级使用技巧和奥秘，使读者能迅速成为 Visual Basic 使用高手。

本书以“问题—解答—实例”的形式编写，针对 Visual Basic 的高级使用技巧进行全面的剖析，其中大部分是许许多多 Visual Basic 程序员梦寐以求的问题解决方案。本书包括 70 个以上的问题及其解决方法和实例，内容涉及窗体、用户界面、控件、多媒体、环境与系统、外设（屏幕、键盘、鼠标、扬声器、串行口）及最后程序的发布等等。同时书中所有实例的源代码都附于随书提供的配套软盘中，读者可以直接使用、编辑和运行它们。

本书适用于 Visual Basic 爱好者、有一定编程经验且想更进一步深入了解 Visual Basic 的程序员以及 Visual Basic 高级用户。

Visual Basic 实用技术精粹

◆ 编　著　萧　枫　尧　远
责任编辑　陈万寿

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
北京密云春雷印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：22.5
字数：566 千字　　1999 年 6 月第 1 版
印数：1－6 000 册　　1999 年 6 月北京第 1 次印刷

ISBN 7-115-07885-8/TP·1155

定价：37.00 元

前　言

Visual Basic 是 Windows 环境下最优秀的程序设计工具之一，如今有大量的程序员在使用 Visual Basic 开发各种类型的 Windows 应用程序。然而，目前深入介绍 Visual Basic 编程技巧的书籍很少，很多书都是对用户手册的改写。

在实用程序的开发过程中，程序员常常会遇到一些难题，例如，如何隐藏和显示任务栏、如何实现任意多边形甚至椭圆形的窗口、如何实现透明的窗口以及如何将应用程序的图标加入任务栏右下角的任务框中等等。

这些难题直接关系到系统的开发进度。令人头痛的是，无论是查书还是从 Visual Basic 的帮助文件都无法获得这些问题的解答。这是因为现在书市上这种类型的书较少，大多数书都是介绍 Visual Basic 的语法，没有深入到具体的问题；同时这些难题的解决很大部分使用到了 Windows API 函数，Visual Basic 的帮助文件对这些函数的功能没有介绍。

针对这种情况，本书将向读者介绍 Visual Basic 高级的编程技巧，其中大部分是许许多多程序员梦寐以求的问题的解决方案。本书以“问题—解答—实例”的形式编写，读者能方便快捷的查找到自己所需要的解决方案。书中所有实例都由 Visual Basic5.0 中文版制作并正确调试通过，且随书提供所有实例的源程序代码，读者可

以直接在自己的应用程序中使用这些代码。

本书主要由萧枫、尧远编写。另外参加本书编写及资料整理的人员还有：韩庆东、吴学红、张长城、韩伟东、司英华、张丽华、万云峰、高迪、张篷飞、凌宇欣、文芳、李秀敏、兰宏志、金向东、刘芳、王麦、王琼、王淼、张朋秋、张朋娟、姚燕、黄晓燕、李慧、莫慧、王玉容、潘锋、陆山等。

由于作者的水平有限，加之时间仓促，书中错误难免，欢迎广大读者批评指正。

作　者

1999 年 3 月

目　　录

第一章 窗体

窗体是 Visual Basic 最基本的核心，是运行应用程序时，与用户交互操作的实际窗口。要想使用 Visual Basic 编写程序，首先要了解窗体的特性。窗体中含有按钮、列表框、图片框以及菜单等控件，正因为这些控件，才使得用 Visual Basic 编写程序成为可能。本章涉及的技巧简单却非常适用，读者可以把它们直接用于自己的运用程序中，如让本窗体始终位于所有窗体之上、实现任意多边形的窗体、实现透明的窗体等等。

由于本书很多技巧的实现都使用到了 API（应用程序接口）函数，所以在每章的开始先列出本章所使用到的 API 函数，以供读者查阅。

本章使用了如下 API 函数：

SetWindowPos	ShellAbout	CreatePolygonRgn
SetWindowRgn	CreateEllipticRgn	SetWindowLong
GetWindowRect	GetWindowLong	GetSystemMenu
RemoveMenu	DeleteMenu	GetMenuItemCount
GetDC	ReleaseDC	SetBkColor
Rectangle	CreateSolidBrush	SelectObject
DeleteObject	ExtractIcon	GetWindowWord
FlashWindow		

本章的主要内容有：

- 让本窗体始终位于所有窗体之上
- 自动调整窗体控件的大小
- 使窗体的标题条闪烁
- 使用 Windows 系统的标准 ABOUT 窗体
- 实现任意多边形的窗体
- 实现椭圆形窗体
- 在运行时隐藏/显示窗体标题栏
- 使窗体的最大化和最小化按钮消失
- 使窗体的关闭、最大、最小化按钮不可用
- 禁止使用 Alt+F4 关闭窗体
- 实现透明的窗体
- 实现可用鼠标切分的窗体
- 使窗口在显示和关闭时出现动态效果
- 窗体事件发生的顺序（窗体小结）

1.1 让本窗体始终位于所有窗体之上

问题

当编写一个字处理软件时，你也许会为如何使查找和替换对话框始终保持在其它窗体之上而冥思苦想；在编写类似于金山词霸等屏幕取词的软件时，你同样需要保持你的窗体在屏幕的最上端。

技巧

要保持窗体在屏幕的最上端，需要用到 SetWindowPos 函数。

函数声明及其使用到的常量声明如下：

```
'SetWindowPos 函数声明
Declare Function SetWindowPos Lib "user32" _
    ( ByVal hwnd As Long, _
    ByVal hWndInsertAfter As Long, _
    ByVal x As Long, ByVal y As Long, _
    ByVal cx As Long, _
    ByVal cy As Long, _
    ByVal wFlags As Long _
    ) As Long
'
'使用的常量声明
Private Const SWP_NOSIZE = &H1
Private Const SWP_NOMOVE = &H2
Private Const HWND_TOPMOST = -1
Private Const HWND_NOTOPMOST = -2
'
```

使用 SetWindowPos 函数和 HWND_TOPMOST 常量就可以使窗体保持在屏幕的最上端，函数的使用方法如下：

```
SetWindowPos myForm.hWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE or SWP_NOSIZE
```

可以将此函数放于窗体的 Load 事件中，把 myForm.hWnd 改为 Me.hWnd，使得窗体在加载时自动保持在屏幕的最上端。

提示：

由于本技巧内容较少，为了节省篇幅，这里不给出具体的例子。本书其它技巧如果没有给出例子，请读者自己试验。

1.2 自动调整窗体控件的大小

问题

当用户调整窗体的大小时，窗体的控件并没有随之放大或缩小，这样在窗体的大小改变后界面会变得很乱。最好的解决方法就是在窗体的 Resize 事件中添加改变控件位置和大小的代码。

技巧

以一个按钮控件为例，假设该按钮控件的名称为 Command1，在窗体的设计状态已经把按钮的位置和大小设置好。要自动调整窗体控件的大小，需要在窗体的 Resize 事件中添加以下代码：

```
Dim fWidth As Integer
Dim fHeight As Integer
Dim cWidth As Integer
Dim cHeight As Integer
Dim cTop As Integer
Dim cLeft As Integer
'
Private Sub Form_Load()
    '获取初始窗体和按钮的大小
    fWidth = Me.ScaleWidth
    fHeight = Me.ScaleHeight
    cWidth = Command1.Width
    cHeight = Command1.Height
    cTop = Command1.Top
    cLeft = Command1.Left
End Sub
'
Private Sub Form_Resize()
    Dim hScale As Double
    Dim vScale As Double
    hScale = ScaleWidth / fWidth
    vScale = ScaleWidth / fHeight
    Command1.Left = cLeft * hScale
    Command1.Top = cTop * vScale
    Command1.Width = cWidth * hScale
    Command1.Height = cHeight * vScale
End Sub
```

1.3 使窗体的标题条闪烁

问题

怎样利用视觉效果来吸引用户的注意，而不采用 Windows 提供的一些“嘈杂”声音？

技巧

要使窗体的标题条闪烁，需要使用 FlashWindow 函数。读者可以在 Timer 事件代码中使用该函数，来制作一个按规律间隔闪烁的窗体。

该函数的声明如下：

```
Declare Function FlashWindow Lib "user32" Alias "FlashWindow" _
    (ByVal hwnd As Long, _
    ByVal bInvert As Long _
    ) As Long
```

函数的使用非常简单，只需把窗体的句柄传递给函数的第一个参数即可，如下所示：

```
Private Sub Timer1_Timer()
    FlashWindow Me.hwnd, 1
End Sub
```

如果要改变标题栏闪烁的速率，可以通过改变 Timer1 的 Interval 属性来实现。

提示

FlashWindow 函数有两个参数：窗体句柄及闪烁标志（称为 bInvert）。窗体句柄由窗体的 hWnd 属性提供；如果 bInvert 标志为非零值，FlashWindow 将闪烁窗体的标题栏，即在激活和非激活状态之间切换。

在窗体最小化成为图标时，FlashWindow 也能工作。

1.4 使用 Windows 系统的标准 ABOUT 窗体

问题

每个程序都有 ABOUT 窗体，用于向用户传达自身的一些基本信息。可以自己设计 ABOUT 窗体，但这里有更简便的方法，即使用 Windows 系统的标准 ABOUT 窗体。

技巧

要使用 Windows 系统的标准 ABOUT 窗体，需要用到 ExtractIcon、GetWindowWord 和 ShellAbout 函数。这些函数的声明以及相关的常量声明如下：

```
' ExtractIcon 函数声明
Declare Function ExtractIcon Lib "shell32.dll" Alias "ExtractIconA" _
```

```
        (ByVal hInst As Long, _
        ByVal lpszExeFileName As String, _
        ByVal nIconIndex As Long _
        ) As Long
    ' GetWindowWord 函数声明
    Declare Function GetWindowWord Lib "user32" _
        (ByVal hwnd As Long, _
        ByVal nIndex As Long _
        ) As Integer
    ' ShellAbout 函数声明
    Declare Function ShellAbout Lib "shell32.dll" Alias "ShellAboutA" _
        (ByVal hWnd As Long, _
        ByVal szApp As String, _
        ByVal szOtherStuff As String, _
        ByVal hIcon As Long _
        ) As Long
    ' 相关的常量声明
    Public Const GWL_EXSTYLE=(-20)
    Public Const GWL_STYLE=(-16)
    Public Const GWL_WNDPROC=(-4)
    Public Const GWW_HINSTANCE=(-6)
```

ShellAbout 函数的使用如下：

```
Ret=ShellAbout (Me.hWnd, "演示程序", _
    "版权所有 [c] 1999-2000 王晟" & Chr$(13) & Chr$(10) & "序列号：123456", Icon)
```

其它两个函数的使用参看实例中的代码注释行。

实例

新建一个工程，在窗体中加入一个按钮。在代码窗口中加入如下代码：

```
' ExtractIcon 函数声明
Private Declare Function ExtractIcon Lib "shell32.dll" Alias "ExtractIconA" _
    (ByVal hInst As Long, _
    ByVal lpszExeFileName As String, _
    ByVal nIconIndex As Long _
    ) As Long
' GetWindowWord 函数声明
Private Declare Function GetWindowWord Lib "user32" _
    (ByVal hwnd As Long, _
    ByVal nIndex As Long _
    ) As Integer
' ShellAbout 函数声明
```

```
Private Declare Function ShellAbout Lib "shell32.dll" Alias "ShellAboutA" _
    (ByVal hWnd As Long, _
    ByVal szApp As String, _
    ByVal szOtherStuff As String, _
    ByVal hIcon As Long _
    ) As Long
' 相关的常量声明
Private Const GWL_EXSTYLE=(-20)
Private Const GWL_STYLE=(-16)
Private Const GWL_WNDPROC=(-4)
Private Const GWW_HINSTANCE=(-6)
' 显示 About 窗口子过程
Public Sub ShowAbout()
    Dim Ret As Long
    Dim Icon As Long
    Dim Inst As Long
    Inst = GetWindowWord(Me.hwnd, GWW_HINSTANCE)
    '从可执行文件中抽取图标
    Icon = ExtractIcon(Inst, "DEMO.EXE", 0)
    '调用 Windows 系统标准 ABOUT 窗口
    Ret = ShellAbout(Me.hwnd, "演示程序", _
    "版权所有 [c] 1999-2000 王晟" & Chr$(13) & Chr$(10) & "序列号: 123456", Icon)
End Sub
' 按钮单击事件
Private Sub Command1_Click()
    ShowAbout
End Sub
```

运行程序，单击窗体中的按钮，运行结果如图 1-1 所示。

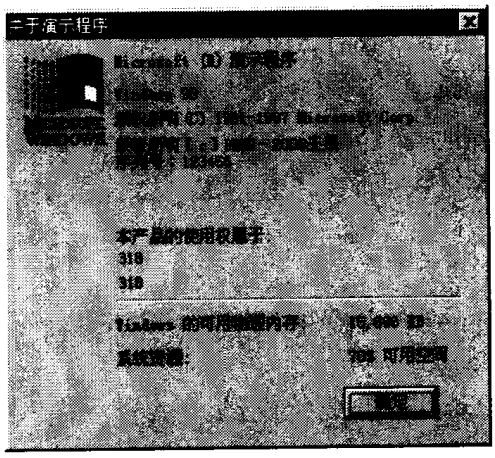


图 1-1　Windows 系统的标准 ABOUT 窗体

1.5 实现任意多边形的窗体

问题

看厌烦了标准的 Windows 方形窗体，你一定想让你的程序多一点创意，那就试验一下建立一个任意多边形的窗体吧。

技巧

建立一个任意多边形的窗体，需要用 CreatePolygonRgn 和 SetWindowRgn 函数。

函数声明如下：

```
' CreatePolygonRgn 函数声明
Declare Function CreatePolygonRgn Lib "gdi32" _
    (lpPoint As POINTAPI, _
    ByVal nCount As Long, _
    ByVal nPolyFillMode As Long _
    ) As Long
' SetWindowRgn 函数声明
Declare Function SetWindowRgn Lib "user32" _
    (ByVal hWnd As Long, _
    ByVal hRgn As Long, _
    ByVal bRedraw As Boolean _
    ) As Long
```

函数的使用如下：

```
Dim hRgn As Long
Dim lRes As Long
Dim XY(7) As POINTAPI
hRgn = CreatePolygonRgn(XY(0), 8, 2)
lRes = SetWindowRgn(Me.hWnd, hRgn, True)
```

其中 CreatePolygonRgn 函数返回值为一个多边形区域的句柄，其各个参数意义为：

1. lpPoint 为 POINTAPI 类型的数组，表示多边形各个顶点的坐标，POINTAPI 类型在 API 声明文件中有如下声明：

```
Type POINTAPI
    x As Long
    y As Long
End Type
```

一个 lpPoint 元素为一个点的坐标，在给 CreatePolygonRgn 函数传递 lpPoint 参数时，只须把 lpPoint 的第一个元素传递过去即可，如上面所演示的使用过程中的参数 XY(0)。

2. nCount 为长整型参数，用来说明多边形的顶点数，如上面所演示的使用过程中的参数 8。

3. nPolyFillMode 为长整型参数，用来指定多边形区域的填充模式，如果参数为 2 则是不透明模式。

SetWindowRgn 函数各个参数意义为：

1. hWnd 为长整型参数，表示要变为多边形窗体的窗体句柄。

2. hRgn 为长整型参数，表示多边形区域的句柄

3. bRedraw 为布尔型参数，表示是否要立刻重新刷新窗体，如果设置为 True 则表示函数调用后立刻刷新。

实例

该例子把窗体变成了一个 T 形。新建一个工程，在窗体中加入一个按钮。在代码窗口中加入如下代码：

```
Private Type POINTAPI
        X As Long
        Y As Long
End Type
Dim XY() As POINTAPI
'
' CreatePolygonRgn 函数声明
Private Declare Function CreatePolygonRgn Lib "gdi32" _
        (lpPoint As POINTAPI, _
        ByVal nCount As Long, _
        ByVal nPolyFillMode As Long _
        ) As Long
' SetWindowRgn 函数声明
Private Declare Function SetWindowRgn Lib "user32" _
        (ByVal hWnd As Long, _
        ByVal hRgn As Long, _
        ByVal bRedraw As Boolean _
        ) As Long
'
' Form_Load 事件代码
Private Sub Form_Load()
        Me.ScaleMode = vbPixels
End Sub
'
'Command1 单击事件的代码
Private Sub Command1_Click()
        Dim hRgn As Long
        Dim lRes As Long
        ReDim XY(7) As POINTAPI 'T 形需要 8 个点
```

```
    With Me
        XY(0).X = 0
        XY(0).Y = 0
        XY(1).X = .ScaleWidth
        XY(1).Y = 0
        XY(2).X = .ScaleWidth
        XY(2).Y = .ScaleHeight / 2
        XY(3).X = .ScaleWidth - (.ScaleWidth / 3)
        XY(3).Y = .ScaleHeight / 2
        XY(4).X = .ScaleWidth - (.ScaleWidth / 3)
        XY(4).Y = .ScaleHeight
        XY(5).X = .ScaleWidth / 3
        XY(5).Y = .ScaleHeight
        XY(6).X = .ScaleWidth / 3
        XY(6).Y = .ScaleHeight / 2
        XY(7).X = 0
        XY(7).Y = .ScaleHeight / 2
    End With
    '
    hRgn = CreatePolygonRgn(XY(0), 8, 2)        ' 返回多边形区域的句柄
    lRes = SetWindowRgn(Me.hWnd, hRgn, True)    ' 设置多边形窗体
End Sub
```

运行程序，单击窗体中的按钮，运行结果如图 1-2 所示。

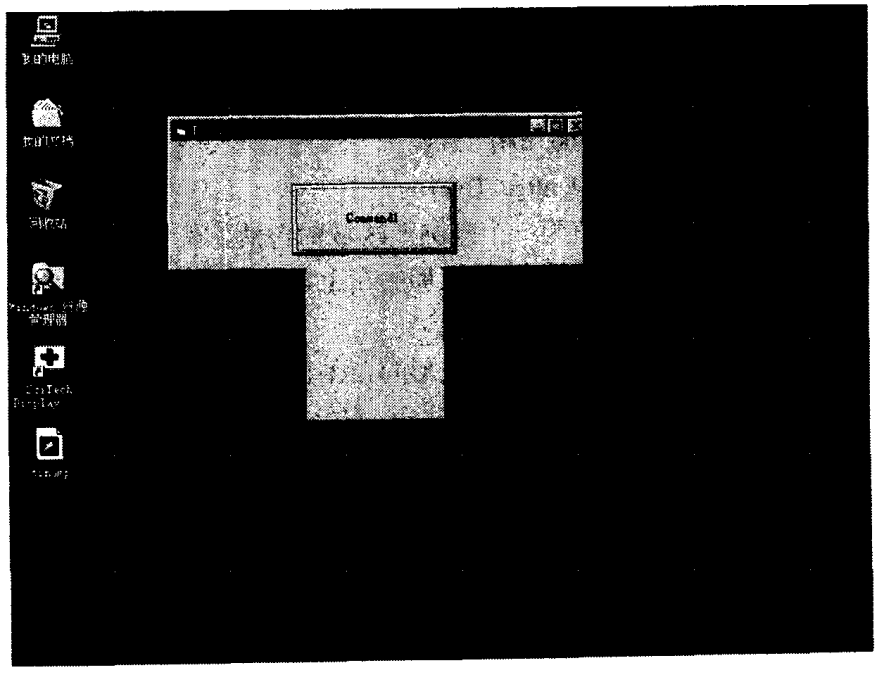


图 1-2 T 形窗体