



Custom Message Boxes in Visual Basic

Visual Basic's MsgBox function is a good one, a direct port from it's API counterpart MessageBox. However, the biggest problem I have with MsgBox is the lack of formatting capability. You're stuck with one font and four icons and you have no control over positioning, size, color or alignment.

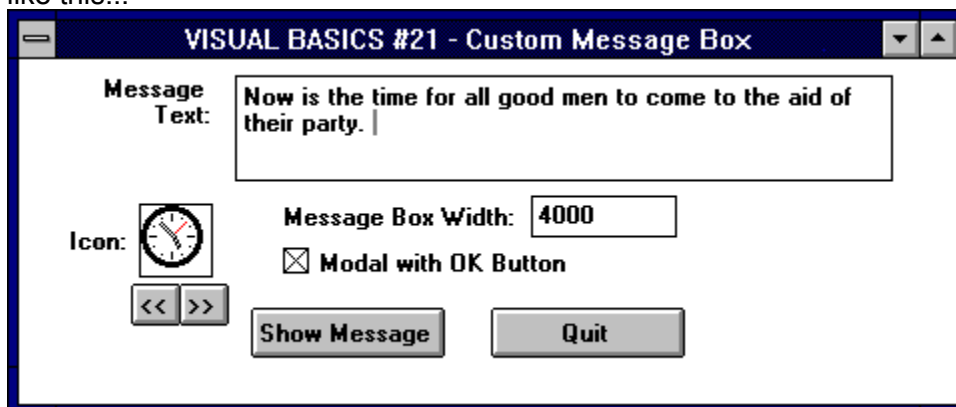
I've created a custom message box which I use for those situations where the standard MsgBox function doesn't have the flexibility I want. With the CustomMsg sub and it's related Message form I now have the capability to control the size of the box, or to put another icon in the place of the four predefined icons we're allowed with the direct call. For example, just try doing **this** with MsgBox...



The functions CustomMsg and WrapText and the form MsgForm use standard VB forms, controls and code, which means you can determine any font characteristics you want for the message text; you control the color and style of the form and title bar and you can add any icon you want to the list of icons available for use with the form.

This week's example is as much useful utility as a how-to file; This ain't simple, and I won't be covering all the code in the example section of this column, so you won't be able to recreate this program by copying code from the sample section to your created forms. However, you **can** download all source code from Windows Online in the file VB021EX.ZIP.

I've included a front end so you can see how the Custom Messagebox is called. It looks like this...



As you can see, you can control a number of attributes of the box, such as...



Message and TitleBar Text



Icon to use (including no icon)



Width of the Box (Height is determined automatically), and



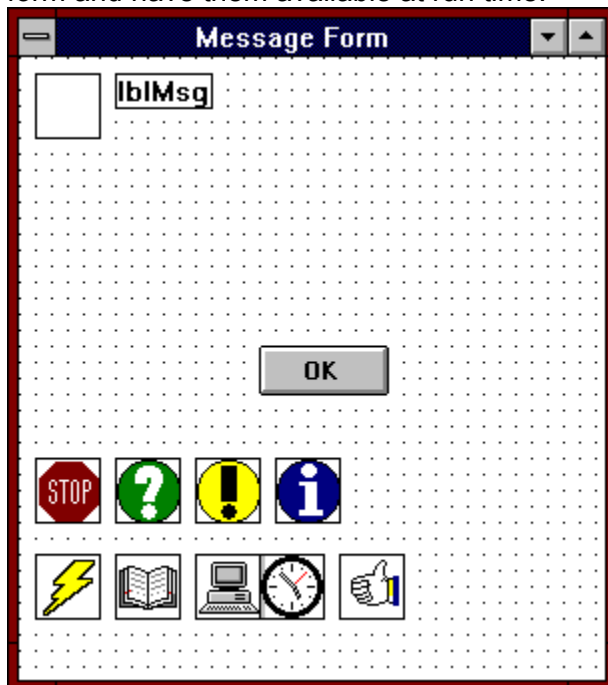
Modality of the Window

In addition, you can control additional attributes directly from code or at design time, including font styles, form colors and everything else.



How it Works...

There's nothing remarkable about the MessageForm itself except for the control array of pictures used to contain available icons. The indexes of these arrays are what you refer to when specifying which icon to use. This allows you to put whatever pictures you want into the form and have them available at run time.



For the sake of both consistency and convenience, I've added pictures with indices of 16,32,48 and 64 and put the standard Windows messagebox icons in them. This helps keep the CustomMsg function consistent with the MsgBox function. The interesting thing to note here is that the array elements are non-contiguous. I've numbered the elements 1,2,3,4,5,16, 32, 48 and 64; Visual Basic doesn't mind the breaks in between. This means you can place other numbered icons on the form as you require and call them by number.



Extra Considerations...

I've included a very involved text justification routine in here called WrapText because I needed to determine the exact size of the label after the message text was placed in it. WrapText works for any control; it evaluates a text string based on the width of the destination control and breaks that text string into individual lines that will fit in the control. It then places each substring into a string array.



Visual Basic Code Sample



```

Do
    SpaceLoc% = InStr(StartPlc%, SourceTxt, " ")
    LFLoc% = InStr(StartPlc%, SourceTxt, LF)
    If SpaceLoc% = 0 And LFLoc% = 0 Then
        NextWord$ = Mid$(SourceTxt, StartPlc%)
    ElseIf SpaceLoc% <> 0 And LFLoc% = 0 Then
        NextWord$ = Mid$(SourceTxt, StartPlc%, SpaceLoc% - StartPlc% +
1)
    ElseIf SpaceLoc% = 0 And LFLoc% <> 0 Then
        NextWord$ = Mid$(SourceTxt, StartPlc%, LFLoc% - StartPlc% + 2)
    ElseIf SpaceLoc% <> 0 And LFLoc% <> 0 Then
        'which comes first? Space or LF?
        If SpaceLoc% < LFLoc% Then 'Space came first...
            NextWord$ = Mid$(SourceTxt, StartPlc%, SpaceLoc% -
StartPlc% + 1)
        Else
            NextWord$ = Mid$(SourceTxt, StartPlc%, LFLoc% - StartPlc%
+ 2)
        End If
    End If

    TabLoc% = InStr(NextWord$, Chr$(9))
    If TabLoc% <> 0 Then
        Lft$ = Left$(NextWord$, InStr(NextWord$, Chr$(9)) - 1)
        Rit$ = Mid$(NextWord$, InStr(NextWord$, Chr$(9)) + 1)
        NextWord$ = Lft$ + Space$(gTabSize) + Rit$
        DebugMsg$ = DebugMsg$ + "TAB Found at " + Format$(TabLoc%,
"0") + LF
    End If

    WordLen% = Len(NextWord$)
    DebugMsg$ = DebugMsg$ + "Word found is [" + NextWord$ + "]" + LF
    DebugMsg$ = DebugMsg$ + "Word Length is " + Format$(WordLen%) + LF

    If DestForm.TextWidth(CreatedTxt(LineQty%) + NextWord$) >
DestCtrl.width Then
        LineQty% = LineQty% + 1
    End If

```

```

CreatedTxt(LineQty%) = CreatedTxt(LineQty%) + NextWord$
StartPlc% = StartPlc% + WordLen%

If StartPlc% >= SourceLength% Then Exit Do
Loop

```

After WrapText formats the text with linefeeds at the appropriate locations, the text is placed in the destination control. Since Label1's AutoSize property is set to TRUE we can now use the label's dimensions to position the icon and set the height of the form accordingly.



How to Use It...

The best example of how CustomMsg is called is in Command3, which is the "Show Message" button, natch...

```

Sub Command3_Click ()
Msg$ = Text1.Text
Titl$ = "Test Message"
FormWidth%=Val(Text2.Text)
If Check1.Value = 1 Then OKBtn% = MODAL Else OKBtn% = MODELESS
CustomMsg Msg$, IconNumber, Titl$, FormWidth%, OKBtn%
End Sub

```

The parameters passed to CustomMsg are as follows:

Msg\$ The message string to use
IconNumber the Index number of the icon picture to use
(Passing zero yields no icon)
Titl\$ The titlebar string
FormWidth% Width of the form in Twips (remember Scalemode is TWIPS)
OKBtn% If **TRUE**, the form is displayed Modally and the OK button is visible. The OK button must be clicked to hide the form.
If **FALSE**, the form is displayed normally with no button.
Focus can be set to other forms, etc. This is handy when you want to pop up a message without hanging the system with a modal form.

Like I said above, there's no example this week; this is a bit too complex to put into an example section you recreate. If you want this code, feel free to download it from Windows Online as VB021EX.ZIP. Feel free to use this message box in any of your applications!

Enjoy!

Barry Seymour

Marquette Computer Consultants

22 Sirard Lane

San Rafael, CA 94901-1066

(415) 459-0835



For **Windows Online "the Weekly"**