

Using ActiveX ProgressBar And ProgressPie Control

1. Overview

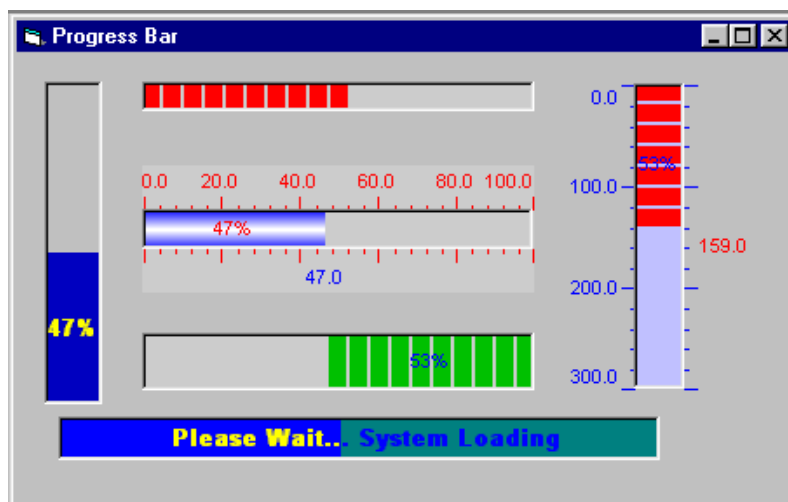
The ProgressBar has been widely used in many applications to show any progress status. We may have experiences that there is a progress bar on the screen when we save some stuff to the disk or load the data from the databases. From the status of the progress bar, we can know status of the ongoing progress (including when the process will be finished, or how long it will take to load or save the required stuff). There is a standard progress bar provided by the Microsoft. But this standard control can not always satisfy our requirements, and its performance is not perfect (flicker problem). These requirements include process bar scale display, percentage display, more active visual look and so on. Therefore for industrial applications or Web applications, we need more powerful process bar ActiveX control.

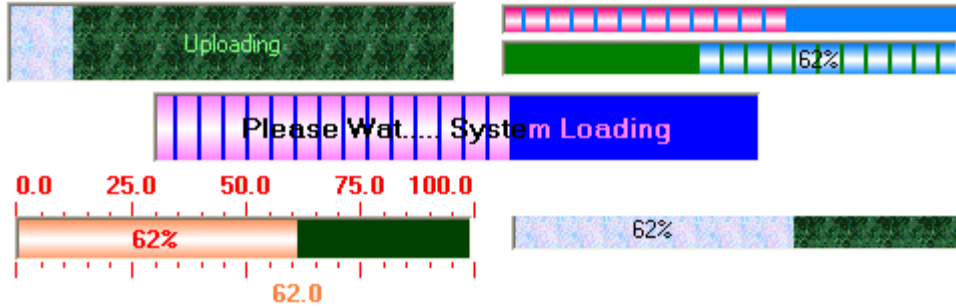
The ActiveX ProcessBar control is designed for many kinds of applications. The interface functions provided by this product are very powerful for designers to configure a great number of kinds of the progress bars. The back ground color, fore color, fonts size, Broken bar style or not, Scale visible or not, Minimum and Maximum, Gradient display or not, Inner percentage display or not, inner text description or not, progress direction, and other features can be configured by the designers at design time or run time. All these properties will be addressed in the following sections.

The ActiveX ProgressPie control is similar to the ActiveX ProgressBar control. The ActiveX ProgressPie uses the circle pie to represent the progress that provides customers another implementation of showing the progress of some actions. Their lightweight feature makes them a good option in the Web design.

2. ProgressBar

The designers can use the interface properties and methods to implement many kinds of the progress bar, the following picture shows some styles which can be configured.





Properties

Property BkColor As OLE_COLOR

This property is used to configure the inner background color of progress bar.

Property BkPicture As IpictureDisp

Defines the standard picture to be rendered in the background of the bar.

Property BrokenProcess As Boolean

If the value of this property is set to *true*, then the progress bar will take form like this

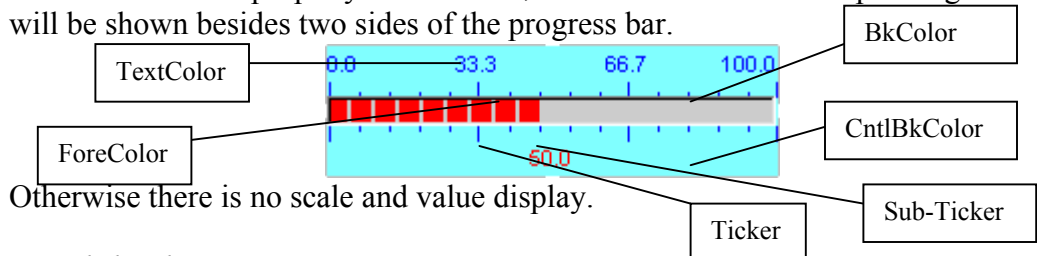


Otherwise



Property bShowValue As Boolean

If the value of this property is set to *true*, then the scale and corresponding the value will be shown besides two sides of the progress bar.



Otherwise there is no scale and value display.

Property CntlBkColor As OLE_COLOR

It is used to set the color of the control background if “*BackStyle = BS_Opaque*”.

Property BackStyle As BackStyleType

If *BackStyle=BS_Opaque*, then the background is rendered using *CntlBkColor*; If *BackStyle=BS_Transparent*, then the background is rendered using the ambient background color of the container of the control.

Property Enabled As Boolean

Specifies whether the window of the ActiveX control is enabled or disabled. If disabled, there is no window message or event triggered by the control.

Property Font As IFontDisp

Defines the standard font property to display the scale texts and the value.

Property ForeColor As OLE_COLOR

This property is used to set the inner foreground color of progress part.

Property Gradient As Boolean

If *bGradient* is set to *true*, then the fore ground color of progress part will be shown in gradient style,



otherwise, it will be shown in normal style.

Property hWnd As Long

Retrieves the window handler of the control. It's a read-only property.

Property ImageStyle As ImagePaintStyle

Specifies the render style of the picture when drawing the picture in the bar area.

There are three definitions,

IPS_ORIGINALSIZE = 0, // the picture is drawn at its original size on the center of bar
// area.

IPS_TEXTURE=1, // the picture is drawn at its original size on the bar area from the
//left-/top point, if the size picture is smaller than the size of the bar area,
//more pictures are drawn to make a texture background which is
//made of multiple copies of the picture.

IPS_AUTOSIZE=2 //the picture will be first resized to the size of the bar area, and then
//drawn on the bar area.

Property InnerPercentage As Boolean

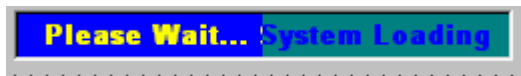
If this property is set to *true*, the actual percentage is display in the inner progress



part. And percentage color is the *textColor*

Property InnerText As String

If *InnerPercentage* is false, and progress bar is horizontal bar, the progress direction is from left to right, then the *InnerText* will show in the whole inner part of progress bar.



Property InverseDirection As Boolean

If the progress bar is horizontal bar, the default progress direction is from left to right, if this property is set to *true*, then the progress direction is from right to left. If the progress bar is vertical bar, the default progress direction is from bottom to top, if this property is set to *true*, then the progress direction is from top to bottom.

Property MaxVal As Double

This property is used to specify the maximum value of the progress bar. If the scale is shown, then the value of this property will be shown on the scale.

Property MinVal As Double

This property is used to specify the minimum value of the progress bar. If the scale is shown, then the value of this property will be shown on the scale.

Property ProgressPicture As IpictureDisp

Defines the standard picture to be rendered in the foreground of the progress bar.

Property SubTickerNum As Integer

This property is used to specify the sub ticker number between two tickers.

Property TextColor As OLE_COLOR

This property is used to specify the color of the scale text color.

Property TickerNum As Integer

This property is used to specify the ticker number on the scale.

Property Value As Double

Set or retrieve the value of the progress bar.

Property Vertical As Boolean

This property is used to specify the progress bar is vertical bar or horizontal bar.

Methods:

Sub UpdateValue(newVal As Double)

This method is used to update the value of the progress bar.

Events

Event Click()

When the chart is clicked, the event is triggered to the container. Fired when the control captures the mouse, any **BUTTONUP** (left, middle, or right) message is received, and the button is released over the control. The **MouseDown** and **MouseUp** events occur before this event.

Event DbClick()

When the chart is double clicked, the event is triggered to the container. Similar to **Click** but fired when a **BUTTONDBLCLK** message is received.

Event KeyDown(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYDOWN** or **WM_KEYDOWN** message is received. *nChar* specifies the virtual key code of the given key. For a list of standard virtual key codes, see **Winuser.h**; *nRepCnt* specifies the repeat count, that is, the number of times the keystroke is repeated as a result of the user holding down the key; *nFlags* specifies the scan code, key-transition code, previous key state, and context code. For details, please refer to MSDN.

Event KeyUp(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYUP** or **WM_KEYUP** message is received. Please refer to *KeyDown* event.

Event *MouseDown(x As Long, y As Long)*

Fired if any **BUTTONDOWN** (left, middle, or right) is received. The mouse is captured immediately before this event is fired. x and y represent the corresponding coordinate values in the control, the origin is at the left-top point of the control.

Event *MouseMove(x As Long, y As Long)*

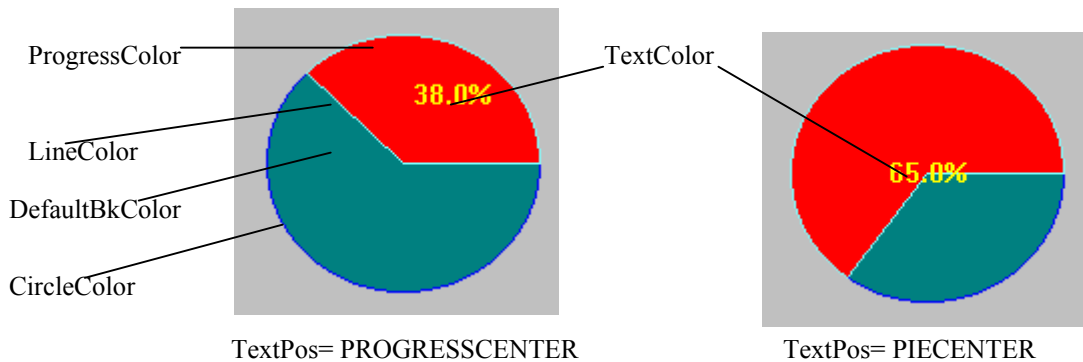
Fired when a **WM_MOUSEMOVE** message is received.

Event *MouseUp(x As Long, y As Long)*

Fired if any **BUTTONUP** (left, middle, or right) is received. The mouse capture is released before this event is fired.

3. ProgressPie

The designers can also use the interface properties and methods of the ProgressPie to implement different styles of the progress status .



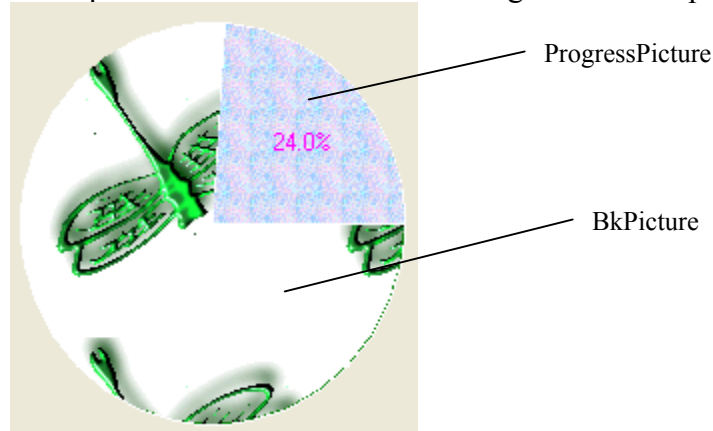
Properties:

Property *BkPicture As IPictureDisp*

Defines the standard picture to be rendered in the background of the pie.

Property *ProgressPicture As IPictureDisp*

Defines the standard picture to be rendered in the foreground of the progress pie.



Property CircleColor As OLE_COLOR

CircleColor is used to specify the color of the circle border.

Property DefaultBkColor As OLE_COLOR

DefaultBkColor is used to specify the default background color of the inner of the circle if “*BackStyle = BS_Opaque*”.

Property BackStyle As BackStyleType

If *BackStyle=BS_Opaque*, then the background is rendered using *DefaultBkColor*; If *BackStyle=BS_Transparent*, then the background is rendered using the ambient background color of the container of the control.

Property Enabled As Boolean

Specifies whether the window of the ActiveX control is enabled or disabled. If disabled, there is no window message or event triggered by the control.

Property Font As IFontDisp

Defines the standard font property to display the scale texts, and the value.

Property hWnd As Long

Retrieves the window handler of the control. It’s a read-only property.

Property ImageStyle As ImagePaintStyle

Specifies the render style of the picture when drawing the picture in the pie area (Please refer to the same property of *ProgressBar*).

Property LineColor As OLE_COLOR

LineColor is used to specify the border color of the progress part.

Property MaxVal As Double

This property is used to specify the maximum value of the progress pie. The control will use the *MaxVal*, *MinVal* and current progress value to calculate the progress percentage.

Property MinVal As Double

This property is used to specify the minimum value of the progress pie. The control will use the *MaxVal*, *MinVal* and current progress value to calculate the progress percentage.

Property ProgressColor As OLE_COLOR

ProgressColor is used to specify the inner foreground color of the progress part.

Property TextColor As OLE_COLOR

TextColor is used to specify the text color of the progress percentage.

Property TextPos As TextPosition

This property is used to specify the position of the percentage text. There are two options we can choose, i.e.,

```
Const PIECENTER = 0 //Text will shown in the center of the control.  
Const PROGRESSCENTER = 1 // Text will shown in the center of the progress part.
```

Methods:

Function UpdatPiecePercentage(Percentage As Double) As Boolean

This method is used to update the progress status (Percentage is the actual value which is used to calculate the final percentage according to *MaxVal* and *MinVal*).

Events

Event Click()

When the chart is clicked, the event is triggered to the container. Fired when the control captures the mouse, any **BUTTONUP** (left, middle, or right) message is received, and the button is released over the control. The **MouseDown** and **MouseUp** events occur before this event.

Event DbClick()

When the chart is double clicked, the event is triggered to the container. Similar to **Click** but fired when a **BUTTONDBLCLK** message is received.

Event KeyDown(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYDOWN** or **WM_KEYDOWN** message is received. *nChar* specifies the virtual key code of the given key. For a list of standard virtual key codes, see *Winuser.h*; *nRepCnt* specifies the repeat count, that is, the number of times the keystroke is repeated as a result of the user holding down the key; *nFlags* specifies the scan code, key-transition code, previous key state, and context code. For details, please refer to MSDN.

Event KeyUp(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYUP** or **WM_KEYUP** message is received. Please refer to *KeyDown* event.

Event MouseDown(x As Long, y As Long)

Fired if any **BUTTONDOWN** (left, middle, or right) is received. The mouse is captured immediately before this event is fired. *x* and *y* represent the corresponding coordinate values in the control, the origin is at the left-top point of the control.

Event MouseMove(x As Long, y As Long)

Fired when a **WM_MOUSEMOVE** message is received.

Event MouseUp(x As Long, y As Long)

Fired if any **BUTTONUP** (left, middle, or right) is received. The mouse capture is released before this event is fired.