

Using ActiveX Percentage Controls

1. Overview

A pie chart is a circle graph divided into pieces, each displaying the size of some related piece of information, which are used to display the sizes of parts that make up some whole. A pie chart is a good way of showing the constituent parts of a variable. It consists of a circle split into segments. The segments represent individual parts which, taken together, make up the total. The 360° circle is divided in proportion to the parts that make up total.

The PieChart and the Multiple percentage bar have been widely used in many application to show percentages in the systems. In financial and statistical analytical fields, the pie chart is a common tool to demonstrate the distributed percentages among different factors, which presents visual and intuitive look and feeling.

A pie chart shows the proportional size of items that make up a data series to the sum of the items. The ActiveX PieChart, PercentageBar and PieChart3D developed by Dragonfly Automation Software are very powerful which can be used in many applications. In the next two sections, we will present the details involving the interfaces of these three kinds components which include functions, properties and methods.

2. PieChart

The ActiveX PieChart Control can show element percentage as a slice of a pie with different color, and the actual percentage value will shown inside each slice. The Pie can be circle or ellipse, which depends on the layout of control instance (if Height=Width, it's a circle; otherwise it's an ellipse). In the following we will discuss all the interface functions, methods and properties.

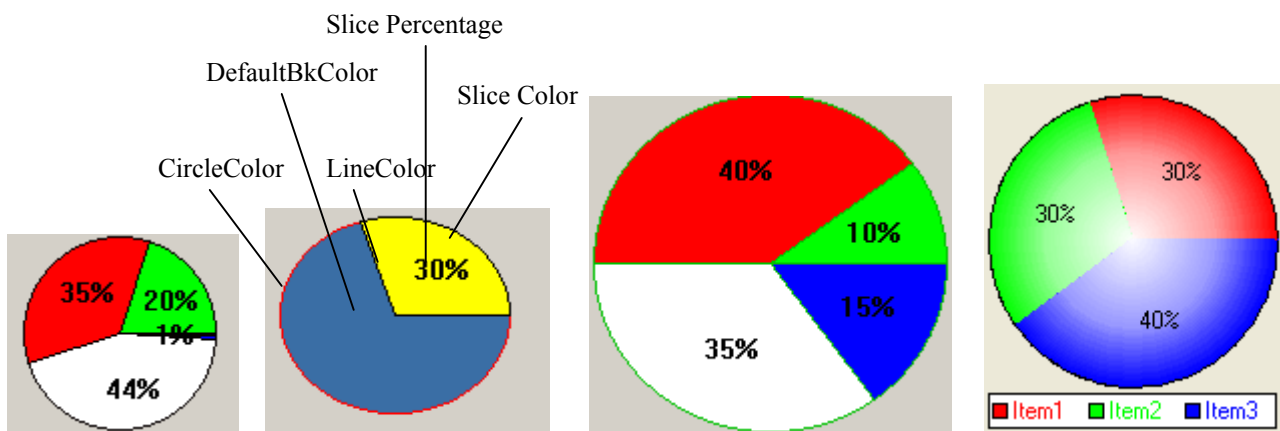


Figure 1

Properties

Property bkColor As OLE_COLOR

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

Property BackStyle As BackStyleType

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

Property *BorderDarkColor* As *OLE_COLOR*

This property is used to configure the color of the border side which is hidden from the light. It is used in conjunction with *BorderLightColor* to create the 3D-Look border.

Property *BorderLightColor* As *OLE_COLOR*

This property is used to configure the color of the border side which faces the light. It is used in conjunction with *BorderDarkColor* to create the 3D-Look border.

Property *InnerBorderLen* As *Integer*

This property is used to specify the length of the inner border.

Property *OuterBorderLen* As *Integer*

This property is used to specify the length of the outer border.

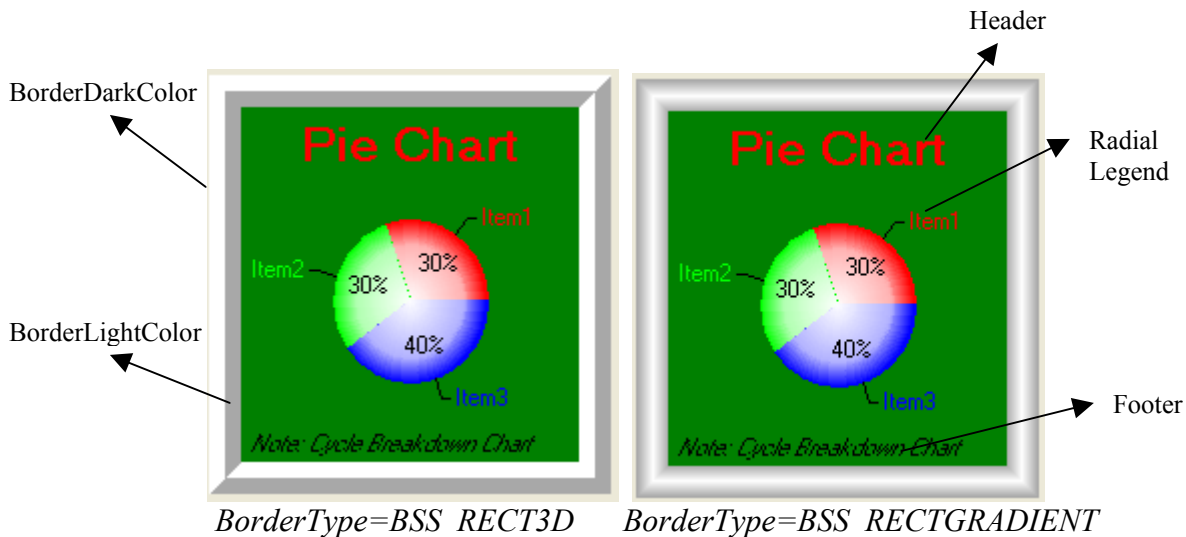


Figure 2

Property *BorderType* As *BorderShowStyle*

Specify the type of the border. Five border types are defined as follows,

BSS_FLAT=0, //No Border

BSS_RECT3D=1, // 3D Rectangle Border

BSS_RECTGRADIENT=2, // 3D Gradient Rectangle Border

BSS_SUNKEN=3, // Sunken Rectangle Border

BSS_RAISED =4 //Raised Rectangle Border

Property *LeftGap* As *Integer*

Property *RightGap* As *Integer*

Property *TopGap* As *Integer*

Property *BottomGap* As *Integer*

All these gap properties are used to adjust the space between the pies and the chart border when the legend is radial (*LegendPos= LPT_RADIAL*, see Figure 2) which can be used to show the radial legend texts.

Property CircleColor As OLE_COLOR

CircleColor is used to specify the color of the pie arc border to draw the entire pie.

Property DefaultBkColor As OLE_COLOR

DefaultBkColor is used to specify the filling color of the entire pie, this filling color will be overwritten by the slice color at each slice part.

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 value texts and legend texts.

Property Footer As String

Defines the text of the footer that is displayed at the bottom of the chart (See Figure 2).

Property FooterColor As OLE_COLOR

Specifies the text color to show the footer text.

Property FooterFont As IfontDisp

Defines the standard font property to display the footer text.

Property ShowFooter As Boolean

Specifies whether to show the footer or not.

Property Header As String

Defines the text of the header that is displayed at the top of the chart. In general, it may be the title of the chart (See Figure 2).

Property HeaderColor As OLE_COLOR

Specifies the text color to show the header text.

Property HeaderFont As IfontDisp

Defines the standard font property to display the header text.

Property ShowHeader As Boolean

Specifies whether to show the header or not.

Property Gradient As Boolean

Specifies whether to draw the piece slice in gradient style or normal style (In Figure 1, the first 3 charts are normal style, and the fourth is gradient style).

Property GradientRate As Single

Define gradient rate for gradient drawing. Its range is from 0 to 1.0, 0 represents no gradient, and bigger value will result in brighter gradient drawing.

Property hWnd As Long

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

Property LineColor As OLE_COLOR

LineColor is used to specify the side border color of each pie slice in non-gradient mode. In gradient mode, it takes no effect.

Property LegendBkColor As OLE_COLOR

Specifies the background color of the legend rectangle box.

Property LegendPos As LegendPosType

Specifies the position of the legend box. There are five legend position types are defined,

LPT_LEFT=0, // Legend is displayed at the left side of the chart.

LPT_RIGHT=1, // Legend is displayed at the right side of the chart.

LPT_TOP=2, // Legend is displayed at the top side of the chart.

LPT_BOTTOM=3 // Legend is displayed at the bottom side of the chart.

LPT_RADIAL=4// Legend is displayed as radial style like Figure 2.

Property ShowLegend As Boolean

Determines whether to show the legend or not.

Property NoPieBorderLine As Boolean

Specifies whether to draw the border of the pie slices or not. If *NoPieBorderLine*=*TRUE*, then *LineColor* and *CircleColor* will not take effect.

Property ShowPercentage As Boolean

Specifies whether to display the actual percentage number of each item on the corresponding pie slice or not.

Property TextColor As OLE_COLOR

Reserved for future use.

Methods & Functions

Sub AddPieChartPiece(Key As String, PieceName As String, bkColor As OLE_COLOR, txtColor As OLE_COLOR, Percentage As Long)

AddPieChartPiece is used to add a new pie slice specified by *Key* to the slice collection of the pie chart, where *PieceName* specifies the name of the new slice, the background color is specified by *bkColor*, percentage text color is specified by *txtColor*, *Percentage* is used to specify the initial percentage of the new slice.

Function *GetPiecePercentage(Key As String, pPercentage As Long) As Boolean*

Retrieve the percentage of the pie item specified by *Key* to the variable pointed by *pPercentage*. If the call is successful, *TRUE* is returned, otherwise *FALSE* is returned.

Sub *Reset()*

Reset all the pie slice settings.

Function *UpdatePiecePercentage(Key As String, Percentage As Long, bViewUpdate As Boolean) As Boolean*

UpdatePiecePercentage is used to update the percentage value of the slice specified by *Key* to the new value *Percentage*, *bViewUpdate* is used to refresh the view of pie chart. If the call is successful, *TRUE* is returned, otherwise *FALSE* is returned.

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 *DblClick()*

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. PercentageBar

The ActiveX PercentageBar Control is similar to PieChart Control which can be used to show element percentage as a piece of a bar with different color, and the actual percentage value will shown inside each piece. The bar can be horizontal or vertical. In the following we will discuss all the interface functions, methods and properties.

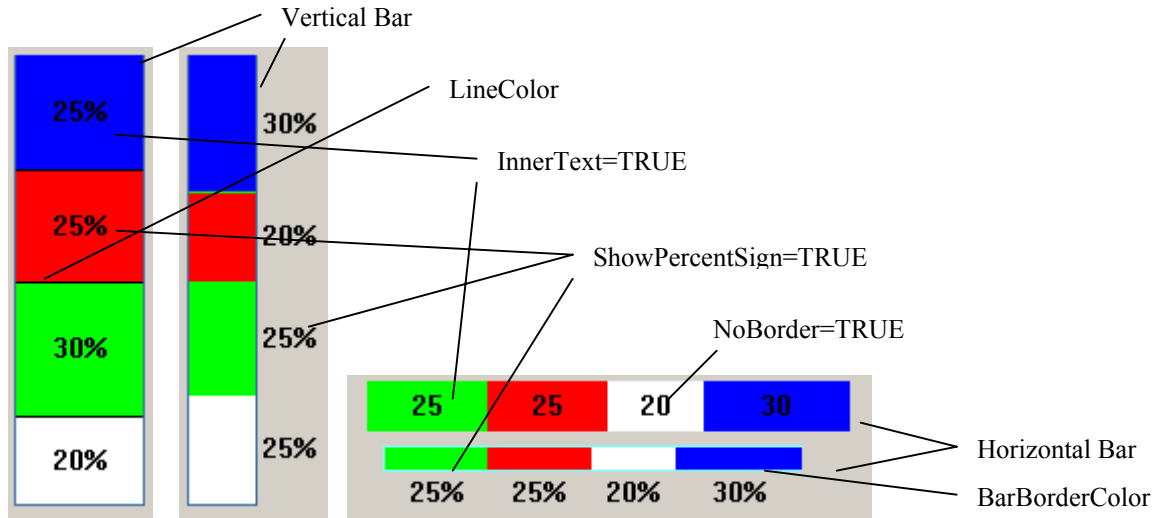


Figure 3

Properties

Property BarBorderColor As OLE_COLOR

BarBorderColor specifies the border color of the bar. Note it is not the color of the chart border, it is the color of the bar border (See Figure 4).

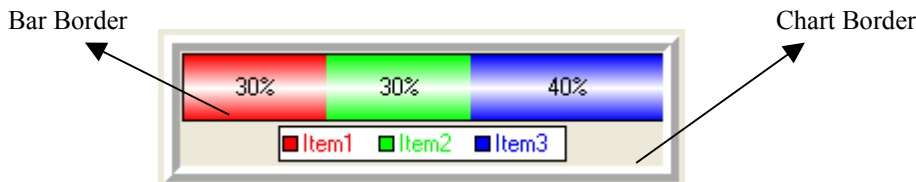


Figure 4

Property bkColor As OLE_COLOR

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

Property BackStyle As BackStyleType

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

Property BorderDarkColor As OLE_COLOR

This property is used to configure the color of the border side which is hidden from the light. It is used in conjunction with *BorderLightColor* to create the 3D-Look border.

Property BorderLightColor As OLE_COLOR

This property is used to configure the color of the border side which faces the light. It is used in conjunction with *BorderDarkColor* to create the 3D-Look border.



Property *InnerBorderLen* As Integer

This property is used to specify the length of the inner border.

Property *OuterBorderLen* As Integer

This property is used to specify the length of the outer border.

Property *BorderType* As *BorderShowStyle*

Specify the type of the border (Please refer to the same property of PieChart in section 2).

Property *DefaultBkColor* As *OLE_COLOR*

DefaultBkColor is used to specify the filling color of the entire bar, this filling color will be overwritten by the bar color at each item part.

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 value texts and legend texts.

Property *Footer* As String

Defines the text of the footer that is displayed at the bottom of the chart .

Property *FooterColor* As *OLE_COLOR*

Specifies the text color to show the footer text.

Property *FooterFont* As *IfontDisp*

Defines the standard font property to display the footer text.

Property *ShowFooter* As Boolean

Specifies whether to show the footer or not.

Property *Header* As String

Defines the text of the header that is displayed at the top of the chart. In general, it may be the title of the chart.

Property *HeaderColor* As *OLE_COLOR*

Specifies the text color to show the header text.

Property *HeaderFont* As *IfontDisp*

Defines the standard font property to display the header text.

Property *ShowHeader* As Boolean

Specifies whether to show the header or not.

Property *Gradient* As Boolean

Specifies whether to draw the bar item in gradient style or normal style.

Property hWnd As Long

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

Property InnerText As Boolean

InnerText is used to determine whether the percentage value of each piece is shown inside the bar or outside the bar. *InnerText* =TRUE for inside display, and *bInnerText* =FALSE for outside display.

Property LegendBkColor As OLE_COLOR

Specifies the background color of the legend rectangle box.

Property ShowLegend As Boolean

Determines whether to show the legend or not.

Property LineColor As OLE_COLOR

LineColor is used to specify the line color of the bar border between two bar items.

Property NoBorder As Boolean

NoBorder specifies whether the bar border is displayed or not. If *NoBorder*=TRUE, then the border of bar will be drawn using *BarBorderColor*, the lines between two bar items are drawn using *LineColor*.

Property ShowPercentage As Boolean

Specifies whether the percentages are displayed or not.

Property ShowPercentSign As Boolean

ShowPercentSign is used to determine whether the “%” is displayed or not.

Property TextColor As OLE_COLOR

Reserved for future use.

Property Vertical As Boolean

Vertical =TRUE, the bar is vertical, otherwise the bar is horizontal.

Property VerticalGradient As Boolean

Specifies whether the gradient direction is vertical or horizontal.

Methods & Functions

Sub AddPercentagePiece(Key As String, PieceName As String, bkColor As OLE_COLOR, txtColor As OLE_COLOR, Percentage As Long)

AddPercentagePiece is used to add a new piece specified by *Key* to piece collection of the PercentageBar chart, where *PieceName* specifies the name of the new piece, the background color is specified by *bkColor*, percentage text color is specified by *txtColor*, *Percentage* is used to specify the initial percentage of the new piece.

Function GetPiecePercentage(Key As String, pPercentage As Long) As Boolean

Retrieve the percentage of the piece item specified by *Key* to the variable pointed by *pPercentage*. If the call is successful, TRUE is returned, otherwise FALSE is returned.

Sub Reset()

All bar items are removed from the bar item collection.

Function UpdatePiecePercentage(Key As String, Percentage As Long, bViewUpdate As Boolean) As Boolean

UpdatPiecePercentage is used to update the percentage value of the piece specified by *Key* to the new value *Percentage*, *bViewUpdate* is used to refresh the view of PercentageBar chart. If the call is successful, *TRUE* is returned, otherwise *FALSE* is returned.

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.



4. PieChart3D

The ActiveX PieChart3D Control is similar to PieChart control, which can show element percentage as a slice of a pie with different color, and the actual percentage value will shown inside each slice. But PieChart3D is developed using OpenGL technique, which presents users real 3D pie chart.

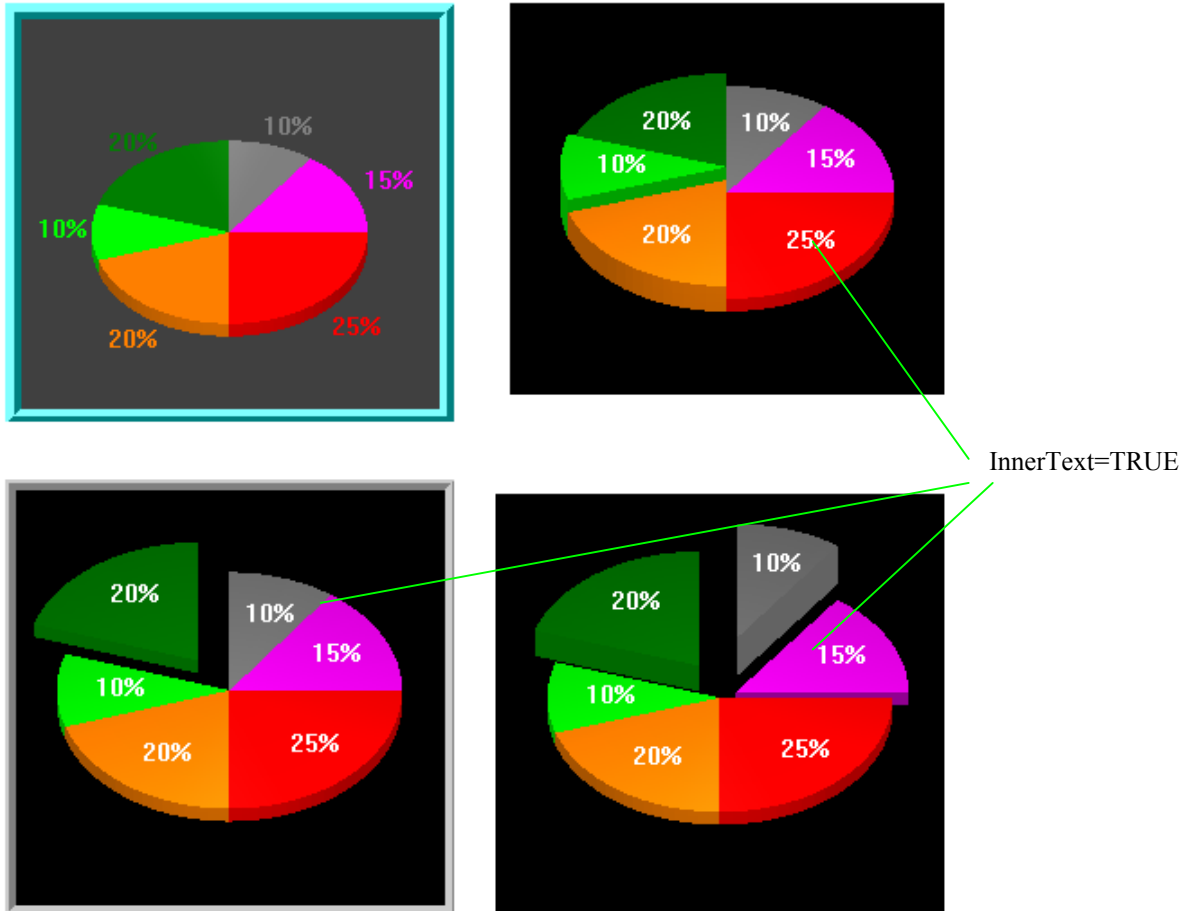


Figure 5

In the following we will discuss all the interface functions, methods and properties.

Properties

Property bkColor As OLE_COLOR

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

Property BackStyle As BackStyleType

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

Property BorderDarkColor As OLE_COLOR

This property is used to configure the color of the border side which is hidden from the light. It is used in conjunction with *BorderLightColor* to create the 3D-Look border.



Property *BorderLightColor* As OLE_COLOR

This property is used to configure the color of the border side which faces the light. It is used in conjunction with *BorderDarkColor* to create the 3D-Look border.

Property *InnerBorderLen* As Integer

This property is used to specify the length of the inner border.

Property *OuterBorderLen* As Integer

This property is used to specify the length of the outer border.

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 *hWnd* As Long

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

Property *InnerText* As Boolean

InnerText is used to determine whether the percentage value of each piece is shown inside the bar or outside the bar. *InnerText* =TRUE for inside display, and *bInnerText* =FALSE for outside display.

Property *ShowPercentage* As Boolean

ShowPercentage specifies whether the percentage value of each slice is displayed or not. TRUE means to show the percentage value, and FALSE means not to show the percentage value See Figure 5, *ShowPercentage*=TRUE; In the Figure 6, *ShowPercentage*=FALSE).

Property *DefaultBkColor* As OLE_COLOR

DefaultBkColor specifies the default background color of the pie slice.

Property *LineColor* As OLE_COLOR

LineColor specifies the border color of each slice.

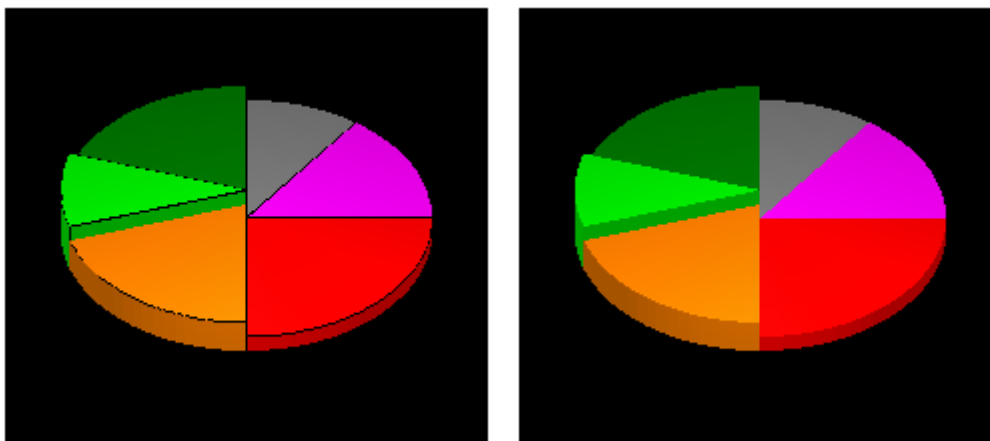


Figure 6

Property NoFrame As Boolean

NoFrame is used to determine whether to show the slice frame (slice border) or not. (See Figure 6, at left chart, *NoFrame*=*FALSE*; at right chart, *NoFrame*=*TRUE*).

Property ObliqueAngle As Integer

ObliqueAngle specifies the angle at any vertical plane at which direction the user looks at the chart. If this property changes, the chart will turn around the horizontal line.

Property Thickness As Integer

Thickness is used to determine the thickness of the pie.

Methods & Functions

Sub AddPieChartPiece(PieceName As String, bkColor As OLE_COLOR, txtColor As OLE_COLOR, Percentage As Long, Thickness As Integer, offset As Integer)

AddPieChartPiece is used to add a new slice to slice collection of the pie chart, where *PieceName* specifies the name of the new slice, the background color is specified by *bkColor*, percentage text color is specified by *txtColor*, *Percentage* is used to specify the initial percentage of the new slice. *Thickness* specifies the thickness of the new slice, which can be different from the default thickness specified by *Thickness* property. Offset is the nearest length between the center point of the pie and the new slice.

Function GetPiecePercentage(PieceName As String, pPercentage As Long) As Boolean

Retrieve the percentage of the piece item specified by *PieceName* to the variable pointed by *pPercentage*. If the call is successful, *TRUE* is returned, otherwise *FALSE* is returned.

Sub Reset()

All bar items are removed from the bar item collection.

Function UpdatPiecePercentage(PieceName As String, Percentage As Long, bViewUpdate As Boolean) As Boolean

UpdatPiecePercentage is used to update the percentage value of the slice specified by *PieceName* to *Percentage*, *bViewUpdate* is used to refresh the view of pie chart. If the call is successful, *TRUE* is returned, otherwise *FALSE* is returned.

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 DblClick()

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.