***Image*** **Sample Help**

**Sample Description:** Image

**Points of Interest**

- Setting the 3D Effects on an Image Control
- Setting the Zoom Factor on an Image
- ScrollBars and the Image Control
- Creating a Bitmap Reference on an Image Control
- Creating an Image Viewer Form
- Activating a Form from the Main Form

**Controls**

- Image
- Bitmap

For Help on Help, Press F1

**Image**

The Image control is designed to display pictures. To display a picture in an image control, you must assign its **Picture** property to some image source, such as a **Bitmap** object. A Bitmap object is responsible for importing the actual graphic picture into the application. When an Image's Picture property references a Bitmap object, and the Bitmap's **FileName** property is set with a valid bitmap file, the image will appear in the Image control.

This sample form is designed to familiarize you with some of the particular properties of an Image control and how they are used. Clicking on the various option buttons on the sample form will set various properties and you will immediately see the results. Clicking on one of the four buttons at the bottom of the form, will change the Image control's Picture property to reference a different Bitmap file. A different Bitmap object is associated with each button's click method.

At the bottom of the sample, is a checkbox. When this checkbox is checked (True), a Bitmap Viewer form will be automatically displayed. From this you can resize to see how the Image control automatically adjusts its size to the size of the form. In addition, you can use the mouse to click and drag a dynamic zoom box for zooming in to the image.

**Setting the 3D Effects on an Image Control**

The **BevelWidth** and **BevelOuter** option buttons will set their corresponding properties on the Image control on the sample. The BevelWidth property is used to set the width of the border around an Image control. The BevelOuter property is used to set the type of border. The following options are available:

| BevelOuter | Description |
|---|---|
| 0 - None | The image control will have no border |
| 1 - Line | The image control will have a line border |
| 2 - Inset | The image control will have a 3D Inset border |
| 3 - Raised | The image control will have a 3D Raised border |

There are other controls that have the BevelWidth and BevelOuter properties such as Forms and PictureBoxes. In addition to BevelOuter, there is a property named **BevelInner** that provides an inner bevel appearance to the form. Other controls, such as TextBoxes, CheckBoxes, and OptionButtons have a **Ctrl3d** property that may be True/False. If set to True, the control will have a 3D appearance.

**Setting the Zoom Factor on an Image**

Setting the zoom factor on an Image is a matter of setting the **CropXOffset**, **CropYOffset**, **CropXSize,** and **CropYSize** factors. In this sample, three zoom factor choices are available, however, all three choices simply send the selected scale factor the following ZoomFactor method.

```
Sub ZoomFactor(factor As Single)
    dim cx, cy as integer
    ' Zoom in on the image by a given scale factor
    imgAttributes.CropXOffset = bitmapAttributes.Width / 2 - (1 / factor) * (bitmapAttributes.Width / 2)
    imgAttributes.CropYOffset = bitmapAttributes.Height / 2 - (1 / factor) * (bitmapAttributes.Height / 2)
    cx = (1 / factor) * (bitmapAttributes.Width)
    cy = (1 / factor) * (bitmapAttributes.Height)
    If (cx < 1) Then cx = 1
    If (cy < 1) Then cy = 1
    imgAttributes.CropXSize = cx
    imgAttributes.CropYSize = cy
    imgAttributes.Refresh
End Sub
```

When a bitmap is fit to an image control, the CropXSize and CropYSize values are set to equal the Width and Height properties of the Bitmap. When a Bitmap FileName is specified, these Width and Height values are automatically set.

When the CropXSize and CropYSize, which represent a viewing region on the bitmap, are smaller than the bitmap width and height, the image will appear zoomed in, or closer. When the CropXSize and CropYSize values are larger than the bitmap width and height, the image will appear zoomed out, or farther away. It is important to understand that the CropXSize and CropYSize automatically fit the Image control boundaries.

The CropXOffset and CropYOffset properties determine offset amounts from the upper left corner of the bitmap, which is the 0,0 origin. Setting the CropXOffset to a value of 10 has the effect of shifting the image 10 pixels to the left. This means that the left side of the bitmap will not be displayed, as it is 10 pixels to the left of the left of the Image control.

**ScrollBars and the Image Control**

The Image control has a ScrollBar property. This property is used to provide scrollbars for panning the display of the bitmap both horizontally and vertically.

| Property Value | Description |
|---|---|
| 0 - Never | The Image controls scrollbars will never be displayed. |
| 1 - Always | The Image controls scrollbars will always be displayed. |
| 2 - Automatic | The Image controls scrollbars will be displayed if needed. |

In the sample form, if the Automatic scrollbar mode is selected, the only time the scrollbars will appear is when the "Double" Zoom factor option is clicked.

**Creating a Bitmap Reference on an Image Control**

In this sample, the Image control shown at the top of the form is named **imgAttributes**. You can click on this control in the Property Editor to view its specific properties. One of the properties shown at the bottom of the property list is named **Picture**. On the left side of the name "Picture" is a chevron symbol (>>) that is used to indicate a "reference" object.

If you click on this property, the property value displayed is ImageMasterForm.bitmapAttributes. This indicates that the images picture source is an object named **bitmapAttributes**. If you look up the property list, you will see this particular bitmap object. If you expand its list of properties, you will see that the FileName property is set to C:\Envelop\Examples\image\ball.bmp. This is the name of the bitmap file that is currently being viewed in the Image control.

At the bottom of the sample, you will see a picture of a traffic light. This too is an Image control that references a bitmap object named **bitmapTrafficLight**. You will notice three buttons labeled Stop, Go, and Caution. These buttons, when clicked, change the FileName property of the bitmapTrafficLight object. When this happens, the Image control **imgTrafficLight** is automatically updated to display the newly loaded bitmap. Several Image controls can be configured to reference the same bitmap source. This saves your application from using unnecessary memory to load more than one copy of the bitmap.

**Creating an Image Viewer Form**

The checkbox at the bottom of the sample form is called "Dynamic Zoom Demonstration." If you click this button to True, a secondary Bitmap View form is automatically displayed. This form contains the following Resize method.

```
Sub Resize()

  ' Prevent the size of the form from being changed
  Width = 3960
  Height = 4515

  ' Keep the size of the image in sync with the size of the form
  imgZoom.Left = 0
  imgZoom.Top = 0
  imgZoom.Width = ScaleWidth
  imgZoom.Height = ScaleHeight

  imgZoom.Refresh
End Sub
```

If you were to use the mouse to drag one of the sides or corners of the form to make it larger or smaller, the Image control would be automatically fit.

**Fitting the Image**

The Bitmap Viewer form has a "Fit" command on the File menu. This command, when clicked, fits the bitmap to the image. The following program code performs this task.

```
imgZoom.CropXOffset = 0
imgZoom.CropYOffset = 0
imgZoom.CropXSize = bitmapZoom.Width
imgZoom.CropYSize = bitmapZoom.Height
imgZoom.Refresh
```

As mentioned earlier, when the Image control's CropXSize and CropYSize are set to the Bitmap's Width and Height, the bitmap is fit inside the image control

**Dynamic Zoom**

Creating the dynamic zoom in this sample is done with the help of a **User32** object's, DrawFocusRect method. The User32 is an interface to a Windows API. The User32 object can be found under the **Globals** object.

The Dynamic zoom is created by three mouse events: MouseDown, MouseMove, and MouseUp as shown below.

```
Sub imgZoom_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
  ' Be sure it's the left_ button
  If (Button And 1) == 0 Then Exit Sub

  ' Set starting corner of box
  X1 = X / 15
  Y1 = Y / 15
End Sub
```

The MouseDown method is designed to capture the starting corner of the bounding box that is to be created. To facilitate this function, properties X1,X2,Y1,Y2 have been added to the form. The MouseMove method is used to track the movement of the mouse pointer and draw a rectangle based on its coordinates.

```
Sub imgZoom_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
  ' Be sure left_ button is depressed
  If (Button And 1) == 0 Then Exit Sub

  ' Erase focus rectangle if it exists
  If X2 Or Y2 Then
    User32.DrawFocusRect imgZoom.hDC, FocusRec
  End If

  ' Update coordinates
  X2 = X / 15
  Y2 = Y / 15
```

```
    ' Update rectangle
    FocusRec.left_ = X1
    FocusRec.top = Y1
    FocusRec.right_ = X2
    FocusRec.bottom = Y2

    ' Adjust rectangle
    If Y2 < Y1 Then Swap FocusRec.top, FocusRec.bottom

    ' Draw focus rectangle
    User32.DrawFocusRect imgZoom.hDC, FocusRec
  End Sub
```

Finally, the MouseUp method is designed to capture the ending X,Y coordinate. Once the starting and ending X,Y coordinates are known and recorded in their respective properties, the CropXOffset, CropYOffset, CropXSize and CropYSize properties of the Image control can be set accordingly.

```
  Sub imgZoom_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
    dim Ret% As Integer,
    dim xs, ys As Single

    ' Be sure it's the left_ button
    If (Button And 1) == 0 Then Exit Sub

    ' Erase focus rectangle if it exists
    If FocusRec.right_ Or FocusRec.bottom Then
       User32.DrawFocusRect imgZoom.hDC, FocusRec
    End If

    ' Set line thickness
    ' Picture1.DrawWidth = 2

    ' Put coordinates in correct order
    If X2 < X1 Then Swap X1, X2
    If Y2 < Y1 Then Swap Y1, Y2

    xs = imgZoom.ScaleX
    ys = imgZoom.ScaleY

    imgZoom.CropXOffset = X1 / xs
    imgZoom.CropYOffset = Y1 / ys
    imgZoom.CropXSize = ((X2 - X1) + 1) / xs
    imgZoom.CropYSize = ((Y2 - Y1) + 1) / ys
    imgZoom.Refresh

    ' Zero the rectangle coordinates
    X1 = 0
    Y1 = 0
    X2 = 0
    Y2 = 0

  End Sub
```

Once the bitmap has been properly zoomed, the starting and ending coordinate properties are automatically cleared and set to 0.

**Activating a Form from the Main Form**

The checkbox at the bottom of the sample form is called "Dynamic Zoom Demonstration." If you click this button to True, a secondary Bitmap View form is automatically displayed. This action occurs as a result of the following method being executed.

```
Sub chkZoomDemo_Click()
    If chkZoomDemo.Value == 0 Then
        chkZoomDemo.Caption = "Off"
        DemoZoomForm.Hide
    Else
        chkZoomDemo.Caption = "On"
        DemoZoomForm.Show
    End If
End Sub
```

This is an example of how one form can be activated from another form. Once the Bitmap Viewer form is displayed on the screen, the user may view the bitmap by using scrollbars.

On the Bitmap Viewer form, an Exit menu command is found under the File menu. When this menu entry is clicked, the following program code is executed.

```
Sub ExitZoom_Click()
    ImageMasterForm.chkZoomDemo.Value = 0
    ImageMasterForm.chkZoomDemo_Click
End Sub
```

In this example, a method on the Bitmap Viewer form is setting a property on another form and then executing a method on that other form. This program code simulates the user manually clicking the checkbox on the first form. With Envelop, it is very easy to write program code to set properties, trigger events, and execute methods between forms.