***Scrollbar* Sample Help**

Sample Description: ScrollBar

Points of Interest

- Working with Colors
- Separate RGB Components of a Color
- Converting Color Decimal to Hexadecimal

Controls

- ScrollBar
- Control

For Help on Help, Press F1

ScrollBar

Scroll bars are common to users of Windows applications. Scroll bars are most frequently used to let the user scroll the contents of a control, such as a ListBox, to bring into view a portion of information outside the visible region.

In Envelop, a scroll bar control consists of a bar with arrows at each end. Between these arrows is a square scroll box or thumb. The scroll box slides along the scroll shaft between the two arrows. You can create both horizontal and vertical scroll bars by setting the **Orientation** property.

A scroll bar can be used as a user input device to specify a value within a prescribed range. By moving the scroll box between the two arrows, the user can specify a value. The scroll bar acts as a sliding input device, similar to the analog slides on audio equipment.

To work with a scroll bar, you must understand the parts of the bar, and their purposes. The scroll box on the scroll shaft indicates the current value specified by the bar. When the user clicks the arrow at either end of the scroll bar, the scroll box moves an incremental unit toward that arrow. This unit corresponds to the value of the ScrollBar's **SmallChange** property. When the user clicks the scroll shaft somewhere between an arrow and the scroll box, it moves a larger incremental unit toward the click position. This unit corresponds to the value of the ScrollBar's **LargeChange** property. Using the mouse a user may also directly click and drag the scroll box along the scroll shaft.

For any scroll bar, you must determine the range of values that the control can designate. The **Min** and **Max** properties are used to specify this range. Min specifies the smallest value that a scroll bar can represent and Max specifies the largest. The ScrollBar's **Value** property is used to represent the current value of the scroll bar. This value should be somewhere between the Min and Max values. If you set the Value property in program code, the scroll bar will automatically update to show the current status.

The first application sample starts with both a Horizontal and Vertical ScrollBar at the top of the form. Clicking and scrolling either of these scrollbars will change the width and height of the Label control.

The second application sample displays six individual scroll bars. Three scroll bars are used to set and control the Red, Green, and Blue values of a colored control. Scrolling any of these scroll bars will change the **BackColor** property of the **ctrlColor** control. Three additional scroll bars are provided to set and control the Hue, Saturation, and Value components of a color. Scrolling any of these scroll bars will also change the BackColor property.

As the various Red, Green, and Blue components of a color are specified by the scroll bars, the converted Hexadecimal color value is displayed directly below the color control. The individual values of Red, Green, Blue, Hue, Saturation, and Value are displayed to the right of their respective scroll bars.

At the bottom of the sample form is a TextBox for entering a Color "decimal" value. By clicking the **Convert** button, the Red, Green, and Blue components of the decimal color are computed and displayed in the three corresponding labels at the bottom of the form.

Working with Colors

Windows supports the full color capability of today's PC equipment. Your Envelop applications can tap these color resources to produce spectacular looking applications.

The BackColor and ForeColor properties control an object's color. Colors are represented by a 4-byte integer value that indicates a mixture of the three electronic primary colors: red, green, and blue. One way to set a color value is to set this number directly. The first byte of this number is ignored. The second, third, and fourth bytes represent how much red, green, and blue will be used in the color mixture. Each color can be assigned a value between 0 and 255 (or 0 to FF in hexadecimal notation), which indicates that color's intensity in the mixture.

You can also assign a color with an RGB function. This function lets you supply a decimal number, between 0 and 255 for each electronic primary color, and returns the hexadecimal (long integer) value that corresponds to an RGB color.

To set the background color of a label to white, click on the label's BackColor property and enter the following value into the Property Editor: RGB(255,255,255)

To set the foreground color (for text) of a label to red, click on the label's ForeColor property enter the following value into the Property Editor: RGB(255,0,0)

Separate RGB Components of a Color

Sometimes when you look at the value of a BackColor or ForeColor property in the Property Editor, a long integer number appears. A color value is a long integer that includes the standard red, green, and blue color components. These separate components are most easily represented with a hexadecimal number. For example, *&Hrrggbb* represents the red components as *rr*, green as *gg*, and blue as *bb*.

At the bottom of the sample is a TextBox labeled "Decimal Value." Enter a color decimal value in this box and click the "Convert" Button to separate the red, green, and blue components of the color. These components will be displayed in a corresponding label at the bottom of the form.

How it works

A combined color value is a long integer that contains the separate red, green, and blue color components packed as a hexadecimal number like &Hrrggbb, you can see that each color component ranges from 0 to 255 (or &H00 to &HFF in hexadecimal) and occupies different positions in the long hexadecimal value. The program code provided in this example computes the red, green, and blue components of the decimal number by using Envelop's bitwise And operator.

The long integer that is the combined color value is packaged as a hexadecimal number. The hex number represents each color in a specific order and in a range from 0-255 (&H00 &HFF in hex). In this sample, Envelop's bitwise And operator is used to compute the RGB components of the decimal.

Converting Color Decimal to Hexadecimal

In this sample scroll bar application, the method **UpdateColor** is used to compute and update the color values. The red, green, and blue values are taken directly from the values of the corresponding scroll bar and converted to a decimal value with the **RGB** function as shown below.

```
' Get RGB scroll bar values
RED = sbrRed.Value
GREEN = sbrGreen.Value
BLUE = sbrBlue.Value

' Convert R, G, and B values to long color value
rgb_value = RGB(RED, GREEN, BLUE)

' Display color in picture box
ctrlColor.BackColor = rgb_value

' Display long color value
lblColorValue.Caption = "H" & Hex(rgb_value)
```

The decimal value is then converted to Hexadecimal form using Envelop's **Hex** function. The result is then displayed in the Label named **lblColorValue**.