

TegoSoft Inc.

Copyright © 1994-1996 TegoSoft Inc. ® All Rights Reserved

P.O.Box 389, Bellmore, NY 11710

Web Site: <http://www.tegosoft.com>

 e-mail: tegosoft@msn.com


Lesson 7 - Shapes


During the course of developing Windows programs, you may need to write programs that display different shapes (e.g., circles, rectangles, squares). In this lesson you'll learn how to write programs that display these shapes.


Create the Directory of the MyShape Program, Start Visual Basic 4, and Save the Form and the Project Files

In this lesson you are going to create several files. So first of all, let's create a directory where the files that you'll create during the course of this lesson are saved.


 Create the **C:\VBMyProg\Lesson07** directory.

 Start Visual Basic, and then select **New Project** from the **File** menu of Visual Basic.

 Select **Project** from the **View** menu of Visual Basic to display the Project window. Click the **View Form** button of the Project window to make **Form1** the selected form.


 While Form1 is the selected window, select **Save File As** from the **File** menu of Visual Basic.

Visual Basic responds by displaying the **Save File As** dialog box.

 Save the Form file as **MyShape.frm** inside the **C:\VBMyProg\Lesson07** directory.


 Select **Save Project As** from the **File** menu of Visual Basic.

Visual Basic responds by displaying the **Save Project As** dialog box.

 Save the project file as **MyShape.vbp** inside the C:\VbMyProg\Lesson07 directory.

Set the Properties of Form1

You'll now set the properties of **Form1**.

 Make the **Form1** window the selected window, and then press the F4 key on your keyboard to display the **Properties** window of Form1.


 Set the properties of **Form1** as follows:


Property	Setting
Name	frmMyShape
Caption	The MyShape Program
Height	3570
Width	3885
BackColor	White

 Select **Save Project** from the **File** menu of Visual Basic to save your work.


Implementing the Exit button

You'll now place the **Exit** button inside the **frmMyShape** form.

 Make sure that the frmMyShape form is the selected window, and then double-click the CommandButton icon inside the Toolbox window.

 Make the CommandButton that you placed inside the form from the selected object, press F4 on your keyboard to display the Properties window, and then set the properties of the CommandButton as follows:

Property	Setting
Name	cmdExit
Caption	E&xit
Height	495
Width	1215
Top	2520
Left	1200

 Double click the **cmdExit** button to display the **Code** window, set the **Object** list box of the Code window to **cmdExit**, set the **Proc** list box of the Code window to **Click**, and then type the following code inside the **cmdExit_Click()** procedure:

```
Private Sub cmdExit_Click()
```

```
End
```


```
End Sub
```

 Select **Save project** from the **File** menu of Visual Basic to save your work.

The code that you typed is executed automatically whenever the user clicks the **Exit** button. This code causes the program to terminate itself.

Placing the Shape Control

You'll now place the **Shape** control inside the frmMyShape form.

 Make sure that the frmMyShape form is the selected form, then double click the **Shape** icon inside the Toolbox window. In Figure 7.1, the icon of the Shape control inside the Toolbox is shown as the icon third row from the bottom and first column from the left. (However, in your Toolbox window, the icon of the Shape control may be located in a different location inside the Toolbox window).

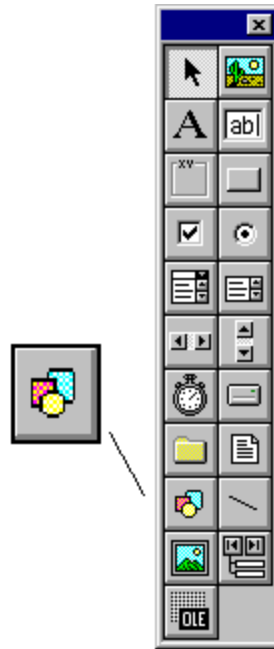


Figure 7.1. *The icon of the Shape control inside the Toolbox window.*



NOTE

When you place the mouse cursor (without pressing any of the mouse buttons) on the Shape icon inside the Toolbox window, a yellow rectangle appears with the text **Shape** in it.

After placing the option button inside the frmMyShape form, the form should look as shown in Figure 7.2.

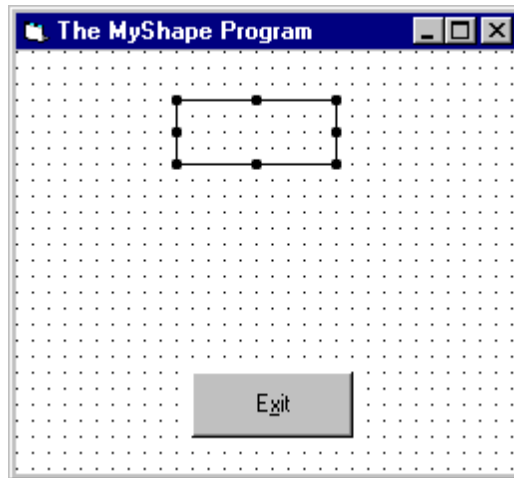



Figure 7.2. The frmMyShape form with the Shape control in it.

Setting the Properties of the Shape Control

You'll now set the properties of the Shape control that you placed inside the form.

 Make the Shape control that you placed inside the frmMyShape form the selected object, press F4 on your keyboard to display the Properties window of the Shape control, and then set the properties of the Shape control as follows:

Property	Setting
Name	shpAnyShape
FillStyle	0-Solid
FillColor	Red
BorderColor	Black
Height	495
Width	1215
Top	360
Left	1200

 Select **Save project** from the **File** menu of Visual Basic to save your work.

Just to make sure that everything is working, let's execute the MyShape program (although you did not yet complete the design of the program):

 Select **Start** from the **Run** menu of Visual Basic.

The window of the MyShape program appears as shown in Figure 7.3.

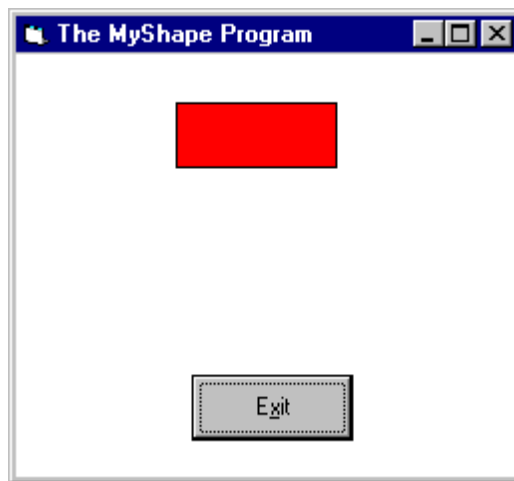



Figure 7.3. *The window of the MyShape program with the Shape control in it.*

Note that the rectangle that appears inside the window of the MyShape program is filled with solid red color. The color is red because you set the **FillColor** property of the Shape control to red. The color is solid because you set the **FillStyle** of the Shape control to 0-Solid.

 Click the **Exit** button of the MyShape program to terminate the program.

Changing the Shape

Take a look at the **Shape** property of the Shape control. Currently, the Shape property is set to **0-Rectangle** (this is the default value that Visual Basic set for the **Shape** property). This is the reason why the Shape control appears as a rectangle. You can set the value of the Shape control to other values. Here are the different shapes that the Shape control can have:

Value of the Shape Property	Resultant Shape
0	Rectangle
1	Square
2	Oval
3	Circle
4	Rounded Rectangle
5	Rounded Square

So to change the shape of the Shape control, you can do one of the following:

- At design time, set the value of the **Shape** property to any integer between 0 and 5.
- At runtime, set the value of the **Shape** property to any integer between 0 and 5. For example, to set the **Shape** property to 3 (circle), use the following statement:


```
' Display a circle
shpAnyShape.Shape = 3
```

Later in this lesson, you'll be instructed to set a new value for the **Shape** property at runtime.

Placing the Width Scrollbar


You'll now place a vertical scrollbar inside the frmMyShape form. This scrollbar will serve as a mechanism for changing the border width of the shape.


=====


 Make sure that the frmMyShape form is the selected window, double click the vertical scrollbar icon inside the Toolbox window, and then press F4 on your keyboard to display the Properties window of the scrollbar that you placed inside the form. Set the properties of the scrollbar as follows:

Property	Setting
Name	vsbBorderWidth
Min	1
Max	10
Value	1
Left	360
Top	1440
Width	255
Height	1215

 Select **Save project** from the **File** menu of Visual Basic to save your work.


 You set the **Min** property of the scrollbar to 1 and you set the **Max** property of the scrollbar to 10. Later in this lesson you'll write code that sets the border width of the shape according to the current position of this scroll bar. Because the scrollbar can have an integer value between 1 and 10, it means that the user will be able to change the border width of the Shape control to a width between 1 and 10.

 In Figure 7.3, the border width of the Shape control is 1. This is because the Shape control has a property called **BorderWidth**, and the default value that Visual Basic assigns to this property is 1.


 Note that the border of the Shape control is painted with black. This is so, because one of the properties that the Shape control has is the **BorderColor** property, and you were instructed to set the BorderColor property of the shpAnyShape control to black.

Placing a Width Label Control

For cosmetic reason, you'll now place a label control above the vsbBorderWidth scrollbar.

 Make sure that the frmMyShape form is the selected form, and then double click the Label control inside the Toolbox.

Visual Basic responds by placing the Label control inside the form.

 Make the Label control that you placed inside the form the selected object, press F4 on your keyboard to display the Properties window, and then set the properties of the Label control as follows:

Property	Setting
Name	lblWidth
Caption	Width
Alignment	2-Center
BorderStyle	1-Fixed Single
BackColor	White
Left	120
Top	1080
Width	735
Height	375

 Select **Save project** from the **File** menu of Visual Basic to save your work.

Your frmMyShape form should now look as shown in Figure 7.4.



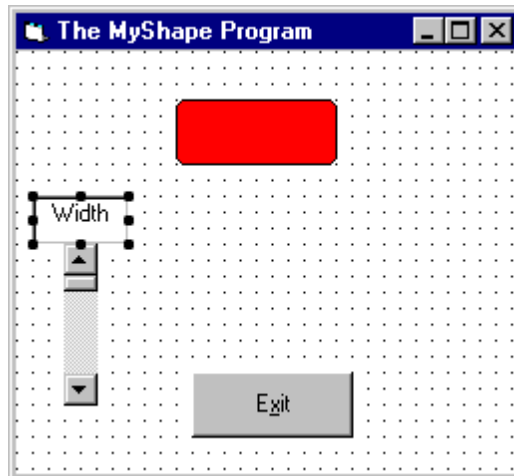



Figure 7.4. The frmMyShape form with its lblWidth label.

Attaching Code to the Width Scrollbar

You'll now attach code to the vsbBorderWidth scrollbar.

 Double click the vsbBorderWidth scrollbar to display the Code window, set the **Object** list box of the Code window to **vsbBorderWidth**, set the **Proc** list box of the Code window to **Change**, and then type the following code inside the vsbBorderWidth_Change() procedure:

Private Sub vsbBorderWidth_Change()

```
shpAnyShape.BorderWidth = vsbBorderWidth.Value
```

End Sub

 Select **Save project** from the **File** menu of Visual Basic to save your work.

The code that you typed is executed automatically whenever the user changes the scrollbar current position. This code sets the **BorderWidth** property of the shpAnyShape Shape control to the **Value** property of the **vsbBorderWidth**

scrollbar. So as the user changes the scrollbar position, the border of the Shape control changes accordingly.

Let's see the **vsbBorderWidth** scrollbar in action:

 Select **Start** from the **Run** menu of Visual Basic.

The window of the MyShape program appears as shown in Figure 7.5.

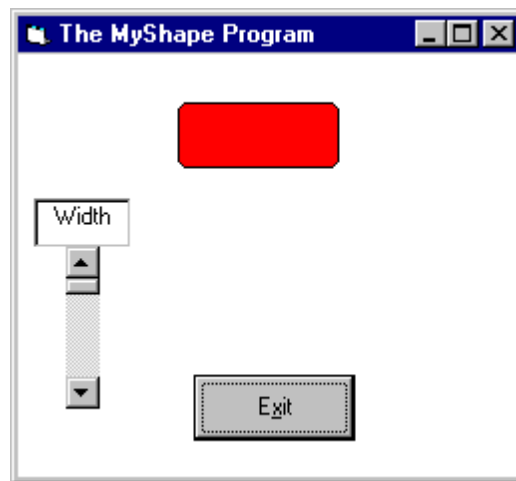



Figure 7.5. *The MyShape program with its vsbBorderWidth scrollbar and its lblWidth label.*

 Change the current position of the scrollbar.

The width of the Shape's border changes accordingly.

For example, Figure 7.6 shows the shape while the vsbBorderWidth scrollbar is at its maximum.

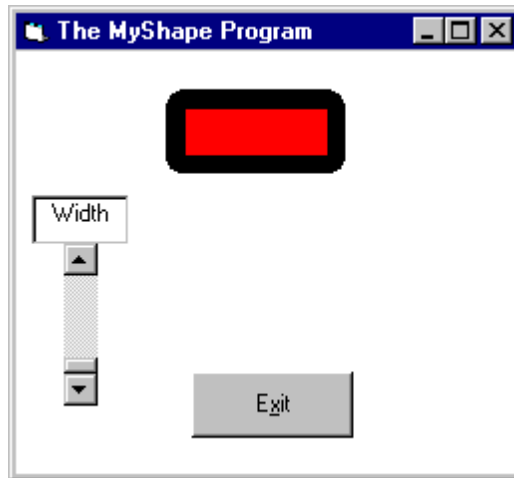




Figure 7.6. *The MyShape program with the vsbBorderWidth scrollbar at its maximum.*

 Experiment with the MyShape program, and then click the **Exit** button to terminate the program.

Attaching Code to the Scroll Event of the BorderWidth Scrollbar

You'll now attach code to the **Scroll** event of the **vsbBorderWidth** scrollbar, so that the border width of the Shape control will change continuously when the user continuously changes the scrollbar position (by dragging the scrollbar thumb).

 Double click the vsbBorderWidth scrollbar to display the Code window, set the **Object** list box of the Code window to **vsbBorderWidth**, set the **Proc** list box of the Code window to **Scroll**, and then type the following code inside the vsbBorderWidth_Scroll() procedure:

Private Sub vsbBorderWidth_Scroll()


```
shpAnyShape.BorderWidth = vsbBorderWidth.Value
```

End Sub

 Select **Save project** from the **File** menu of Visual Basic to save your work.

The code that you typed is executed automatically whenever the user changes the scrollbar position by dragging the thumb of the scrollbar. This code sets the **BorderWidth** property of the **shpAnyShape** Shape control to the **Value** property of the **vsbBorderWidth** scrollbar. So as the user continuously changes the scrollbar position, the border width of the Shape control will change accordingly. (The code inside the `vsbBorderWidth_Scroll()` procedure is identical to the code that you typed inside the `vsbBorderWidth_Change()` procedure).


Let's see the **vsbBorderWidth** scrollbar in action:

 Select **Start** from the **Run** menu of Visual Basic to start the program.

The window of the MyShape program appears as shown back in Figure 7.5.


 Change the scrollbar position by dragging the scrollbar thumb.

The width of the border of the Shape control changes continuously as you change the scrollbar position.

 Experiment with the MyShape program, and then click its **Exit** button to terminate the program.

Placing the Shape Scrollbar

You'll now place a second scrollbar inside the `frmMyShape` form. This scrollbar will serve as a mechanism to change the shape. (The Shape control can display various shapes. The scrollbar that you are about to place will let the user select a shape).

 Make sure that the `frmMyShape` form is the selected form, and then double click the vertical scrollbar icon inside the Toolbox window. Press

=====


F4 on your keyboard to display the Properties window of the scrollbar, and set the properties the scrollbar as follows:

Property	Setting
Name	vsbShape
Min	0
Max	5
Value	0
Left	3000
Top	1440
Width	255
Height	1215

 Select **Save project** from the **File** menu of Visual Basic to save your work.

Placing a Shape Label Control

For cosmetic reason, you'll now place a label control above the vsbShape scrollbar.

 Make sure that the frmShape form is the selected window, double click the Label control inside the Toolbox window, and then press the F4 key on your keyboard to display the Properties window of the Label control. Set the properties of the Label control as follows:


Property	Setting
Name	lblShape
Caption	Shape
Alignment	2-Center
BorderStyle	1-Fixed Single

BackColor	White
Left	2760
Top	1080
Width	735
Height	375

 Select **Save project** from the **File** menu of Visual Basic to save your work.

Attaching Code to the Shape Scrollbar

You'll now attach code to the vsbShape scrollbar.

 Double click the vsbShape scrollbar to display the Code window, set the **Object** list box of the Code window to **vsbShape**, set the **Proc** list box to **Change**, and then type the following code inside the vsbShape_Change() procedure:

Private Sub vsbShape_Change()

```
shpAnyShape.Shape = vsbShape.Value
```

End Sub

 Select **Save project** from the **File** menu of Visual Basic to save your work.

The code that you types is executed automatically whenever the user changes the current position of the **vsbShape** scrollbar. This code sets the Shape property of the **shpAnyShape** Shape control to the **Value** property of the scrollbar. Because you previously set the **Min** property of the scrollbar to 0 and the **Max** property of the scrollbar to 5, the **Shape** property can be any integer

=====

between 0 and 5. So as the user changes the vsbShape scrollbar position, the corresponding shape is displayed.

Let's see this in action:

 Select **Start** from the **Run** menu of Visual Basic.

 Change the current position of the vsbShape scrollbar.

As you change the scrollbar position, the shape that is being displayed by the Shape control changes accordingly. Figures 7.7 through 7.11 show the window of the MyShape program at various settings of the vsbShape scrollbar.

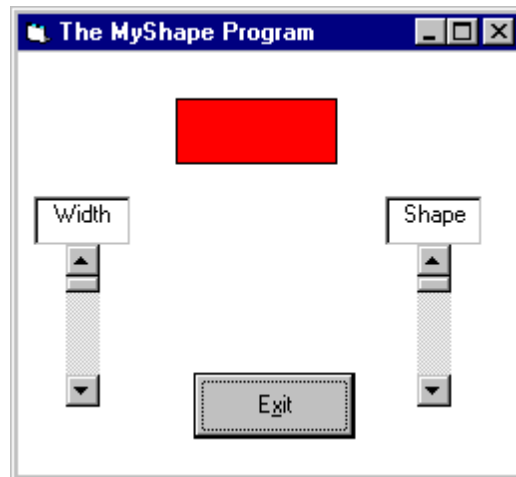


Figure 7.7. *The Value property of the vsbShape scrollbar is set to 0 (Rectangle).*

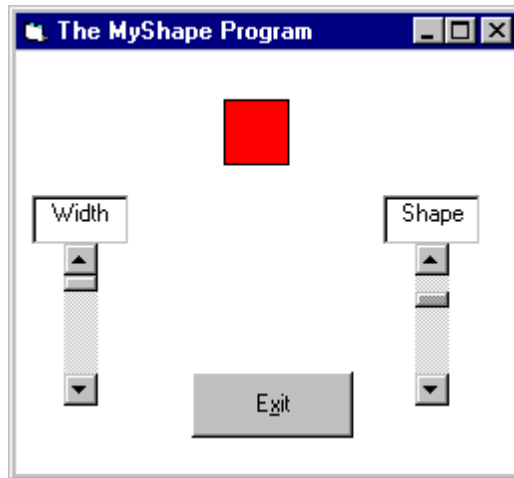


Figure 7.8. *The Value property of the vsbShape scrollbar is set to 1 (Square).*

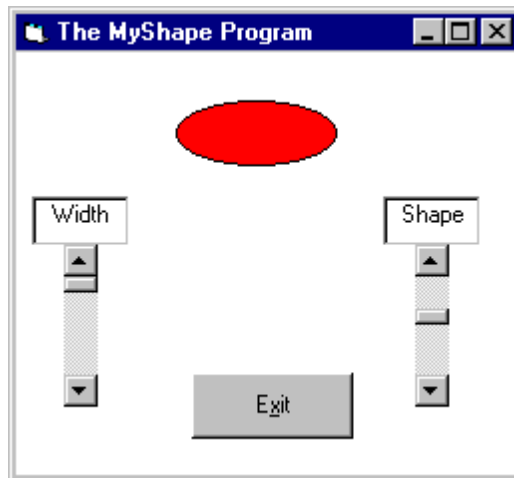


Figure 7.9. *The Value property of the vsbShape scrollbar is set to 2 (Oval).*

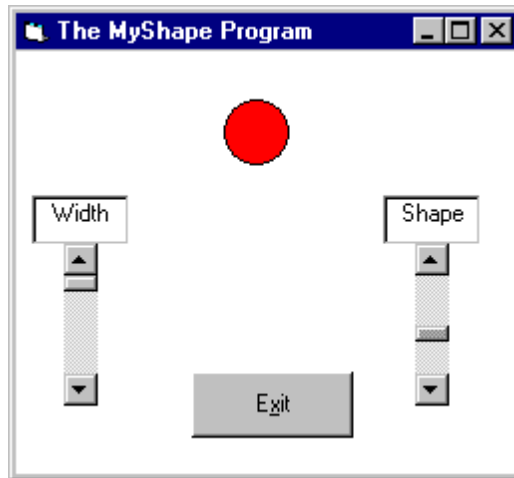


Figure 7.10. *The Value property of the vsbShape scrollbar is set to 3 (Circle).*

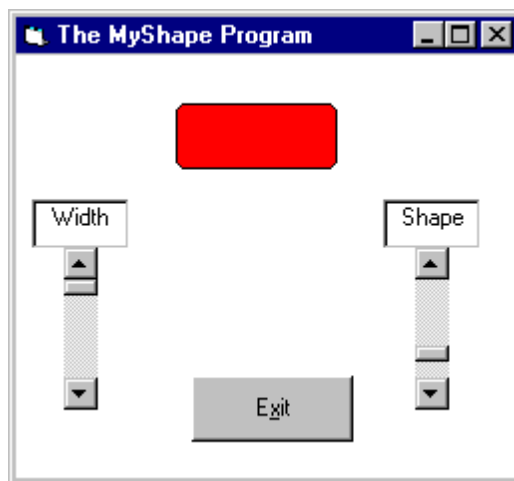


Figure 7.11. *The Value property of the vsbShape scrollbar is set to 4 (Rounded rectangle).*

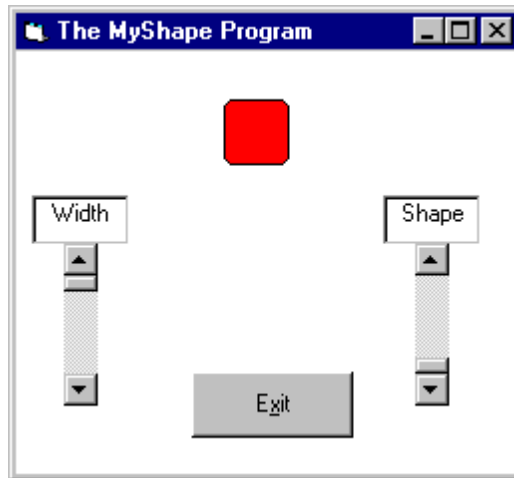




Figure 7.12. *The Value property of the vsbShape scrollbar is set to 5 (Rounded square).*

 Experiment with the MyShape program, and then click its **Exit** button to terminate the program.

Attaching Code to the Scroll Event of the Shape Scrollbar

To be able to change the shape continuously as the user drags the thumb of the scrollbar, you have to attach code to the **Scroll** event of the vsbShape scrollbar.

 Double click the vsbShape scrollbar to display the Code window, set the **Object** list box of the Code window to **vsbShape**, set the **Proc** list box of the Code window to **Scroll**, and then type the following code inside the vsbShape_Scroll() procedure:

```
Private Sub vsbShape_Scroll()
```

```
    shpAnyShape.Shape = vsbShape.Value
```

```
End Sub
```


The code that you typed is executed automatically whenever the user drags the thumb of the scrollbar. As the user drags the thumb of the scrollbar, the shape changes continuously.

 Select **Save project** from the **File** menu of Visual Basic to save your work.

 Select **Start** from the **Run** menu of Visual Basic.

 Change the current position of the **vsbShape** scrollbar.

As you change the scrollbar position by dragging the thumb of the scrollbar, the Shape control displays the corresponding shape accordingly.

 Experiment with the MyShape program, and then click its **Exit** button to terminate the program.

What You Accomplished in This Lesson



You completed Lesson 7 of the Self Study Visual Basic tutorial.


The MyShape program that you implemented in this lesson demonstrates how to display different shapes with the **Shape** control of Visual Basic.


Frequently Asked Questions



Q1. One of the properties of the Shape control is the **FillStyle** property. The Properties window of the Shape control lets me set the **FillStyle** property to 1-Transparent. What does it mean to set the **FillStyle** property of the Shape control to 1-Transparent?

A1. When the **FillStyle** property of the Shape control is set to 1-Transparent, the "inside" of the Shape control is transparent. That is, when you place the Shape control over a BMP picture for example, the area that is covered by the transparent area of the Shape control will be displayed. To see this in action, performed the following experiment:

 Start a new project in Visual Basic (no need to save the new form, and no need to save the new project).

 Set the **Picture** property of Form1 to a BMP picture. For example, set the **Picture** property of Form1 to the **Beany.BMP** file that comes with Visual Basic.

The Beany.BMP file comes with Visual Basic. For example, if you installed Visual Basic inside the directory:

C:\Program Files\Microsoft Visual Basic

then you'll find the Beany.BMP file inside the directory:

C:\Program Files\Microsoft Visual Basic\bitmaps\assorted

Alternatively, use Paint (or Paintbrush) to draw your own small BMP picture, and then set the **Picture** property of Form1 to the BMP picture that you created.

Figure 7.13 shows Form1 after you set its **Picture** property to the Beany.BMP file.

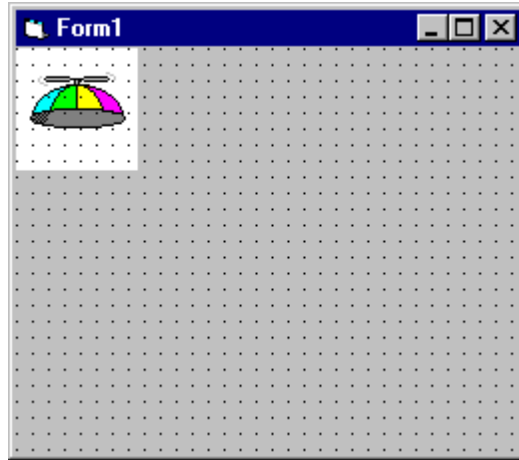


Figure 7.13. *Form1* after setting its *Picture* property to *Beany.BMP*.



NOTE

Note that once you set the **Picture** property of Form1 to a certain BMP picture, the picture appears with its upper left corner in the upper left corner of form (as you can see for example in Figure 7.13).



Place a Shape control inside Form1. Make sure that the **FillStyle** property of the Shape control that you placed inside Form 1 is set to 1-Transparent.



Drag the Shape control that you placed inside the form so that it is partially covers the BMP picture. See Figure 7.14, which shows the Shape control covers a portion of the Beany.BMP picture.

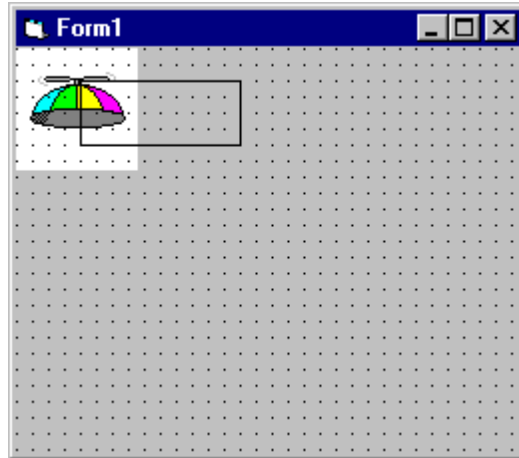


Figure 7.14. *The Shape control is placed over the BMP picture (during design time).*

As you can see from Figure 7.14, the Shape control is transparent. That is, the portion of the BMP that is covered by the Shape control is displayed. (The Shape control is like a transparent glass, because you set the **FillStyle** property to 1-Transparent.) Also, because the background color of Form1 is light gray, this color is shown through the portion of the Shape control that covers the light gray area.

Figure 7.15. shows the resultant window after you execute the program.

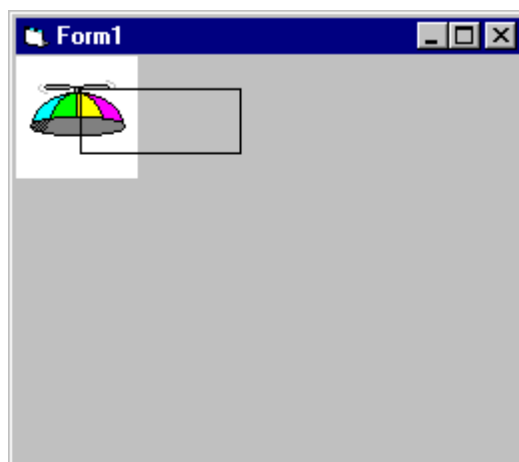


Figure 7.15. *The Shape control is placed over the BMP picture (during runtime time).*

In short, the **FillStyle** property of the Shape control makes the inside of the shape transparent.

Exam



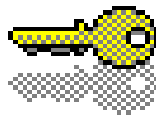
1. The Shape control can be programmed to display a Circle.

- (a) True
- (b) False

2. The Shape control can be programmed to display a triangle.

- (a) True
- (b) False

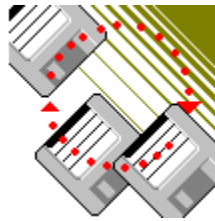
Answers to Exam



1. (a) True

2. (b) False

Project



Modify the MyShape program so that instead of filling the shape with solid red color, the shape is filled with red diagonal cross lines (as shown for example in Figure 7.16).

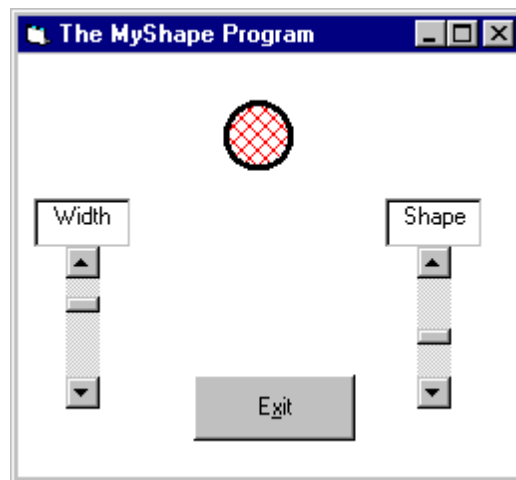



Figure 7.16. *The Shape control is filled with red diagonal cross lines.*


As it turns out, accomplishing the preceding modification is very easy:

 Set the **FillStyle** property of the shpAnyShape Shape control to **7-Diagonal Cross**.

 Select **Start** from the **Run** menu of Visual Basic.

The Shape control is now filled with red diagonal cross lines. (Red is used because you previously set the **FillColor** property of the Shape control to

red, and diagonal cross lines are used because you set the **FillStyle** property of the Shape control to **7-Diagonal Cross**).

 Experiment with the MyShape program by setting different values for the two scrollbars. Then click the **Exit** button to terminate the program.

How To Contact TegoSoft



You can contact TegoSoft Inc. by any one of the following methods:

- Use TegoSoft Internet Web site:

<http://www.tegosoft.com>

- Send TegoSoft an e-mail:

tegosoft@msn.com

- Send TegoSoft a letter:

TegoSoft Inc.

P.O.Box 389

Bellmore, NY 11710

USA

Technical Support



If you have a technical question, you can post the question to the TegoSoft Technical Support staff, and they will try to answer your question.

The **best** way to post a technical question is by sending TegoSoft an e-mail.

● The e-mail of TegoSoft is:
tegosoft@msn.com

When sending TegoSoft an e-mail with a technical question, please follow the following format:

Date: _____
Your name: _____
Company (if applicable): _____
Your phone number: _____
Your e-mail: _____
Country (if not USA): _____
State (if inside USA): _____

Operating System used: _____
Programming language and version : _____

My technical question is:

Copyright © and Notices

Copyright © 1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

TegoSoft Self Study Tutorials & Software

Copyright ©1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

Although precaution was taken in the preparation of this document, TegoSoft assumes no responsibility for errors or omissions. TegoSoft is not liable for damages resulting from the use of the information contained in this document. Use this document at your own risk. This document is copyright protected. This means that you should treat this document like any other copyright material. No part of this document should be copied in

=====

any way. You are not allowed to use this document or any part of this document for any purpose other than read it. You are not allowed to publish this document or any part of this document in any way. You are not allowed to sell this document or any part of this document, you are not allowed to incorporate this document or any part of this document in any book, magazine, Web Site, Electronic forums, disks, CDs, or any other media. The only thing that you are allowed to do with this document is read it for the sole purpose of studying the material that is presented in this document. If this document includes software, the software must be treated in the exact same way that this document is treated. You are not allowed to distribute the software in any way. The sole purpose of supplying the accompanying software is to enable you to experience with it. No part of this document should be modified. If accompanying software is included with this document, you are not allowed to modify the software.

Rev. 031796-1