

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 27 -

Using The Clipboard to Copy, Past and Cut Pictures


In this lesson you'll learn to write programs that utilize the Clipboard to copy, paste and cut pictures.


Creating the Directory of the Project and Saving the Files of the Project


As usual, you'll start by creating the directory of the project, and saving the files of the project.

 Create the **C:\VBMyProg\Lesson27** directory.


You'll save the files of this lesson to this directory.

 Select **New Project** from the **File** menu of Visual Basic.

 Make the window of Form1 the selected window, select **Save File As** from the **File** menu of Visual Basic, and save the file as **ClipPic.FRM** inside the **C:\VBMyProg\Lesson27** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **ClipPic.VBP** inside the **C:\VBMyProg\Lesson27** directory.

Always Use Option Explicit

 Inside the general declarations section of Form1 type the following code:

```
' Force variable declarations  
Option Explicit
```

The **Option Explicit** statement forces you to declare variables before using the variables.

Designing the Window of the ClipPic Program

You'll design the window of the ClipPic program so that it will look as shown in Figure 27.1.

 Set the properties of Form1 as follows:

Name: Form1

Caption: The ClipPic Program

BackColor: White

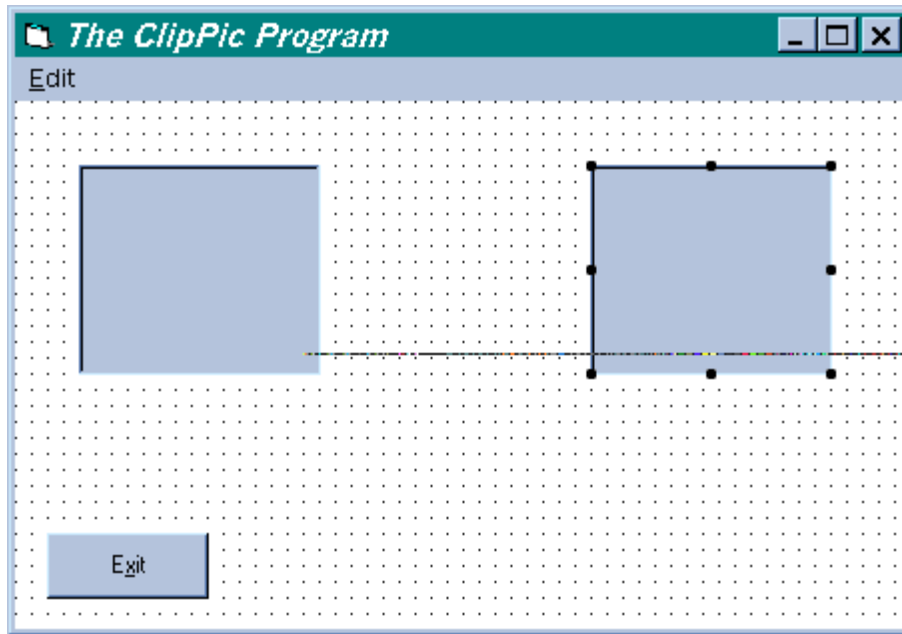



Figure 27.1. Form1 of the ClipPic program

Implementing the Exit Button

You'll now implement the Exit button:


 Place a CommandButton inside Form1, then set the properties of the CommandButton as follows:

Name: cmdExit

Caption: E&xit

Attaching Code to the Click Event of the Exit Button

You'll now attach code to the **Click** event of the Exit button:

 Add the following code inside the **cmdExit_Click()** procedure:

```
Private Sub cmdExit_Click()
```

=====


End

End Sub

So whenever the user clicks the Exit button, the ClipPic program terminates itself.

Placing Picture Controls


You'll now place two Picture controls inside Form1.

 Double click the **Picture** control inside the Toolbox window of Visual Basic.

Visual Basic responds by placing a Picture control inside Form1.


 Drag and move the Picture control to the left (see Figure 27.1).

 Make sure that the **Name** property of the Picture control that you placed inside Form1 is **Picture1**.

 Again double click the **Picture** control inside the Toolbox window of Visual Basic.


Visual Basic responds by placing a second Picture control inside Form1.

 Drag and move the Picture control to the right (see Figure 27.1).

 Make sure that the **Name** property of the Picture control that you placed inside Form1 is **Picture2**.

Implementing the Edit Menu

You'll now attach a menu to the ClipPic program.

 Make sure that Form1 is the selected window, select **Menu Editor** from the **Tools** menu of Visual Basic, and then implement the following menu:

Caption	Name	Shortcut
&Edit	mnuEdit	
...&Copy	mmuCopy	Ctrl+C
...&Paste	mnuPaste	Ctrl+V
...Cu&t	mnuCut	Ctrl+X

Your **Edit** menu should look as shown in Figure 27.2.

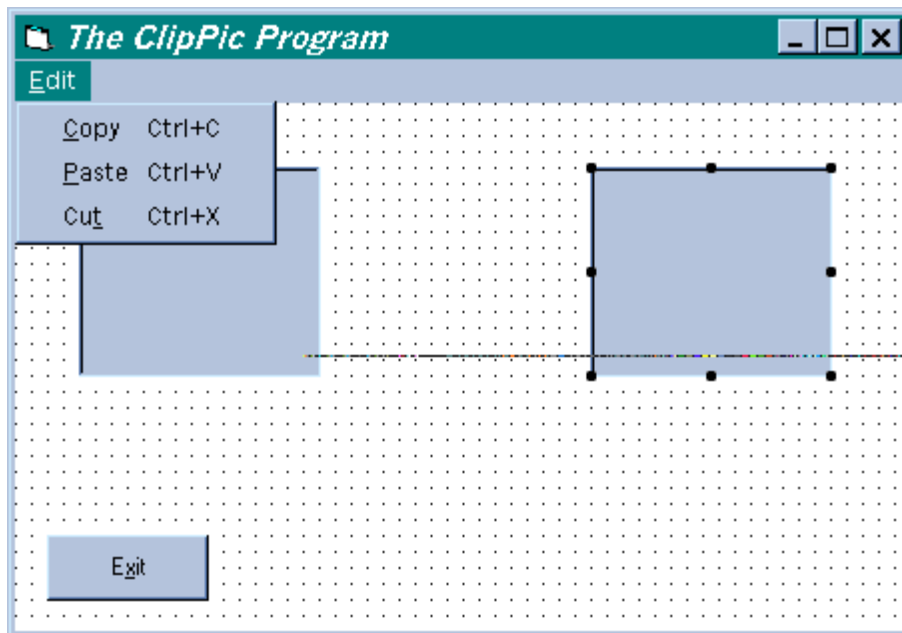



Figure 27.2. The Edit menu of the ClipPic program.

Implementing the Paste Mechanism

You'll now implement the Paste mechanism.

 Type the following code inside the **mnuPaste_Click()** procedure:

```
Private Sub mnuPaste_Click()  
  
If TypeOf Screen.ActiveControl Is PictureBox Then  
  
    Screen.ActiveControl.Picture = Clipboard.GetData()  
  
End If  
  
End Sub
```

The code that you typed uses an **If** statement to determine if the active control is the Picture control:


```
If TypeOf Screen.ActiveControl Is PictureBox Then
```

If the Picture control is the active control, the code under the **If** statement is executed:


```
Screen.ActiveControl.Picture = Clipboard.GetData()
```


The preceding statement uses the **GetData()** method to copy the data that currently resides inside the Clipboard into the Picture control.

Let's see your code in action:

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

 Execute the ClipPic program.

 Start the Paint (or Paintbrush) program that comes with Windows.

 Set the size of the image in Paint (or Paintbrush) to 100 pixels by 100 pixels. (For example, in Paint you set the size of the image by selecting **Attributes** from the **Image** menu of Paint to display the **Attributes** dialog box. Inside the **Attributes** dialog box select the **Pels** radio button to indicate the units of measurements as pixels. Then type **100** in the **Width** text box and type **100** in the **Height** text box. Click the OK button to close the **Attributes** dialog box).

 Draw something in Paint (or Paintbrush). (See for example Figure 27.3).




 Save the picture that you created as **MyPic.BMP** inside the **C:\VBMyProg\Lesson27** directory.





Figure 27.3. *Drawing a picture in Paint.*

 Select **Select All** from the **Edit** menu of Paint (or Paintbrush) and then select **Copy** from the **Edit** menu of Paint (or Paintbrush).

Paint (or paintbrush) responds by copying the picture to the Clipboard.

 Switch back to the ClipPic picture.

 Click inside one of the Picture controls.

 Press Ctrl+V on your keyboard (or select **Paste** from the **Edit** menu of the ClipPic program) to paste the picture that currently resides inside the Clipboard.

As a result, you see the picture that you created in Paint (or Paintbrush) copied into the Picture control of the ClipPic program (see Figure 27.4).

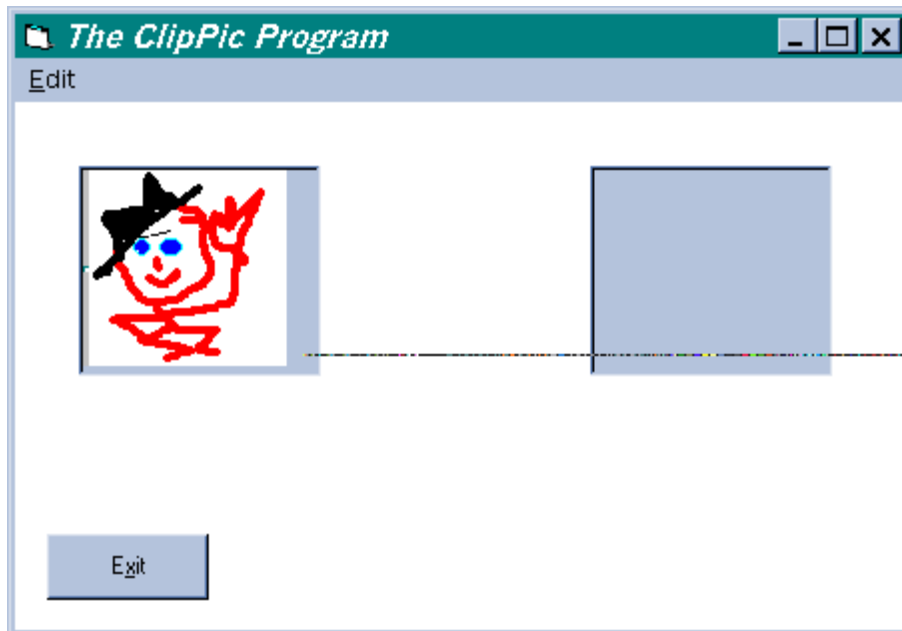




Figure 27.4. *Pasting the picture that you created with Paint (or Paintbrush) into the Picture control of the ClipPic program.*

 Experiment with the ClipPic program, then click the Exit button to terminate the program.

Setting the Picture Property of the Picture Control at Design Time

Before going on with the implementation of the ClipPic program, let's set the **Picture** property of the Picture1 picture control:

 At design time, set the **Picture** property of the Picture1 control to the **MyPic.BMP** picture that you saved inside the **C:\VBMyProg\Lesson27** directory.

Form1 should now look as shown in figure 27.5.

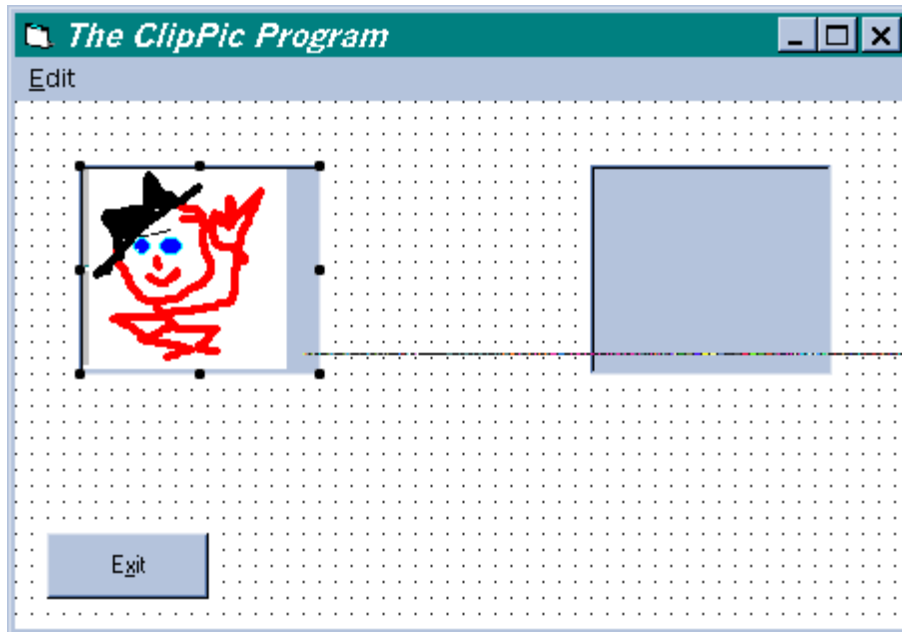




Figure 27.5. *Form1 at design time (after setting the Picture property of the Picture1 picture control to the MyPic.BMP picture).*

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

 Execute the ClipPic program, and verify that the MyPic.BMP picture appears inside the Picture1 control.

 Click the Exit button to terminate the program.




NOTE

You assigned the MyPic.BMP picture to the Picture1 picture control at design time. This means that the MyPic.BMP file becomes an integral part of the program. You can select **Make EXE File** from the **File** menu of Visual Basic, and create the ClipPic.EXE file. If you decide to distribute the ClipPic.EXE file, you do **not** have to distribute the MyPic.BMP file together with the ClipPic.EXE file. Why? Because the MyPic.BMP file is an integral part of the ClipPic.EXE file.

Implementing the Copy Mechanism

You'll now implement the Copy mechanism.

 Type the following code inside the **mnuCopy_Click()** procedure:

```
Private Sub mnuCopy_Click()
```

```
If TypeOf Screen.ActiveControl Is PictureBox Then
```

```
    Clipboard.Clear
```

```
    Clipboard.SetData Screen.ActiveControl.Picture
```

```
End If
```


```
End Sub
```

The code that you typed checks if the active control is the Picture control, and if so, the clipboard is cleared, and then the contents of the Picture control is copied to the Clipboard.

Let's see your code in action:


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

 Execute the ClipPic program.

 Click inside the Picture1 picture control to make the control the active control, and then press Ctrl+C (or select **Copy** from the **Edit** menu).

The ClipPic program responds by copying the MyPic.BMP picture that resides inside the Picture1 picture control to the Clipboard.

=====

 Click inside the Picture2 picture control to make the Picture2 control the active control, and then press Ctrl+V on your keyboard (or select **Paste** from the **Edit** menu).

The ClipPic program responds by pasting the Clipboard contents into the Picture2 picture control. So now Picture2 contains the MyPic.BMP picture (see Figure 27.6).

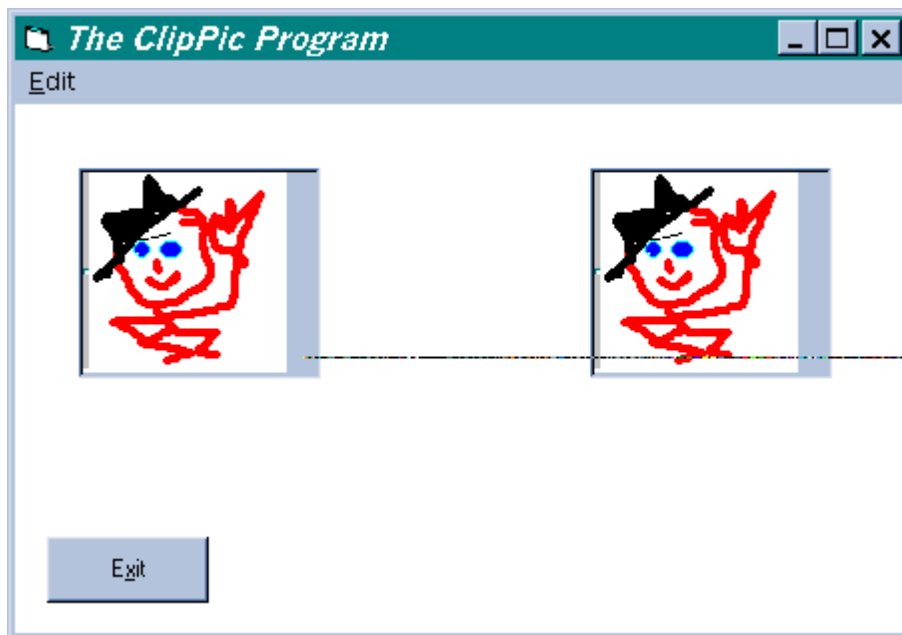



Figure 27.6. Copying the contents of the Picture1 control to the Picture2 control.

 Experiment with the ClipPic program, then click the Exit button to terminate the program

Implementing the Cut Mechanism

You'll now implement the Cut mechanism.

 Type the following code inside the `mnuCut_Click()` procedure:

```
Private Sub mnuCut_Click()
```

```
If TypeOf Screen.ActiveControl Is PictureBox Then
```

```
    mnuCopy_Click
```

```
    Screen.ActiveControl.Picture = LoadPicture()
```

```
End If
```

```
End Sub
```

The code that you typed checks if the active control is the Picture control, and if so, the Copy operation is performed, and finally, the Picture control is filled with null:


```
Screen.ActiveControl.Picture = LoadPicture()
```

Note how the **Picture** property is set in the preceding statement. The **Picture** property of the active control is set to the returned value of the **LoadPicture()** function. Because you did not supply any parameter to the **LoadPicture()** function, the **LoadPicture()** function returns null.

Let's see your code in action:

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

 Execute the ClipPic program.

 Click inside the Picture1 picture control and then press Ctrl+C on your keyboard (or select **Cut** from the **Edit** menu).

The ClipPic program responds by copying the picture that resides inside the Picture control to the Clipboard, and then the contents of the Picture control is set to null. (See Figure 27.7)

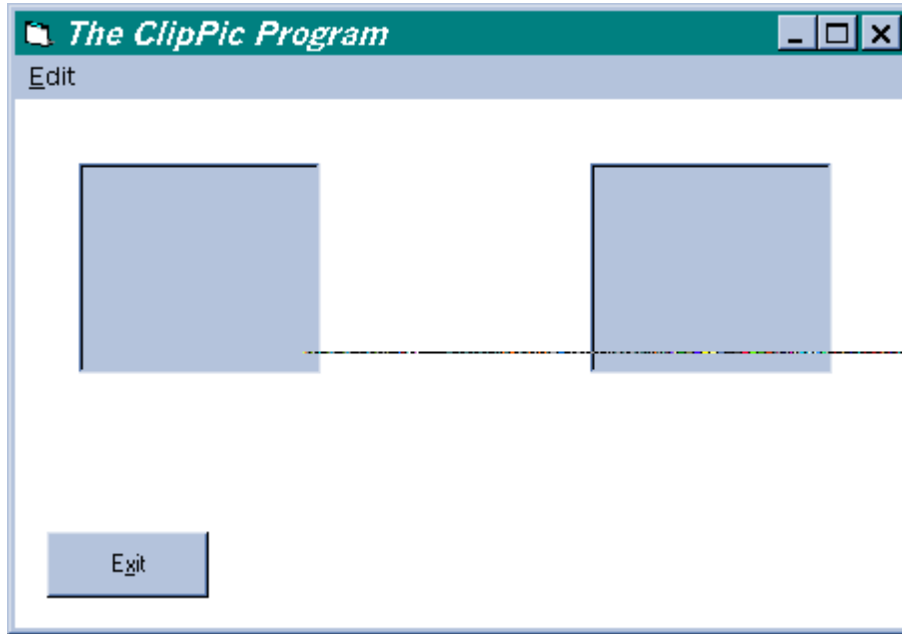



Figure 27.7. *Deleting the picture from the Picture1 control.*

 Click inside the Picture2 control, and press Ctrl+V on your keyboard (or select **Paste** from the **Edit** menu).

The ClipPic program responds by pasting the contents of the Clipboard to the Picture2 control (see Figure 27.8).

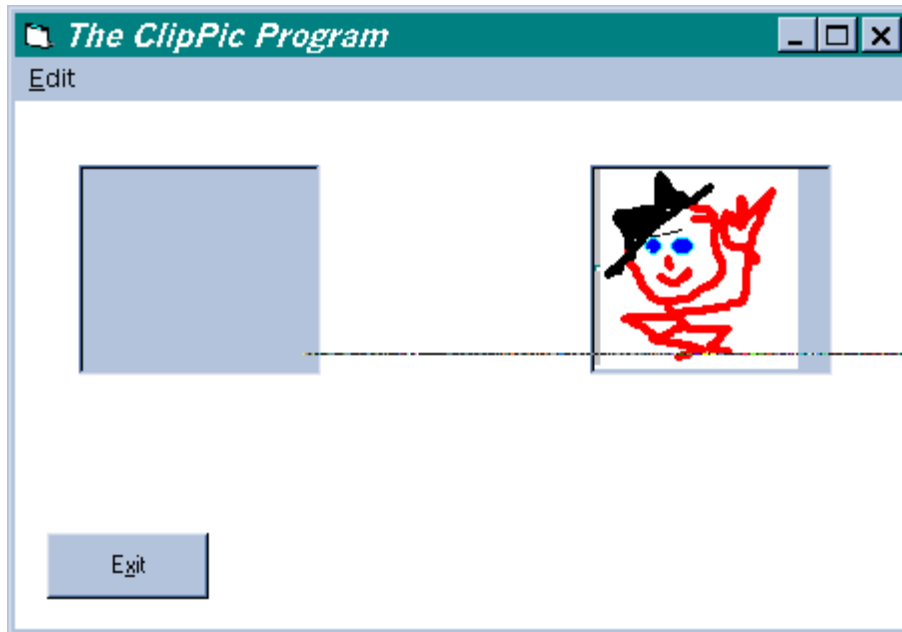



Figure 27.8. *Pasting the picture to the Picture2 control.*

 Experiment with the ClipPic program, then click the Exit button to terminate the program

What You Accomplished in This Lesson



You completed Lesson 27 of the Self Study Visual Basic tutorial. In this lesson you learned to write programs that perform the standard Clipboard copy, Paste and cut operations. In the Cosmetic Considerations section of this lesson you'll learn about the important **GotFocus** and **LostFocus** events.

Frequently Asked Questions




Q1. Suppose that I also have a text box control inside Form1. Will an error occur if my user will copy the picture into the Clipboard and then will




try to paste the picture into the text box? (That is, a text box can contain text, not pictures. Yet, my user will try to paste a picture into the text box!).

A1. No error will occur.

Do the following to verify this:

 Place a Text box control inside Form1.

 Modify the **mnuPaste_Click()** procedure so that it will look as follows:

```
Private Sub mnuPaste_Click()
```

```
  If TypeOf Screen.ActiveControl Is PictureBox Then
```


```
    Screen.ActiveControl.Picture = Clipboard.GetData()
```

```
  ElseIf TypeOf Screen.ActiveControl Is TextBox Then
```

```
    Screen.ActiveControl.SelText = Clipboard.GetText()
```

```
  End If
```

```
End Sub
```


 You added an **Elseif** statement that checks if the active control is the text box:


```
ElseIf TypeOf Screen.ActiveControl Is TextBox Then
```

If the active control is the text box, the **GetText()** method is executed on the Clipboard, and the text of the Clipboard is copied to the text box.

In other words, you should think of the Clipboard as composed of two separate entities:

- One part of the Clipboard is dedicated for storing text. You get the text from the Clipboard by using the **GetText()** method.
- Another part of the Clipboard is dedicated for storing other type of data (non-text). You get the text from the Clipboard by using the **GetData()** method.

 Execute the ClipPic program, copy the contents of the Picture control to the Clipboard, then try to paste the contents of the Clipboard (which currently contains picture) to the text box. As you can see, no error is produced.

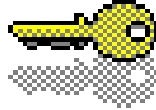
 Click the Exit button to terminate the program.

Exam



1. To set the Picture property of a Picture control to null during runtime you have to: _____

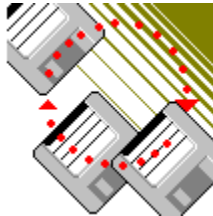
Answers to Exam







1. To set the Picture property of a Picture control to null during runtime you have to use the **LoadPicture()** function as follows:

```
Picture1.Picture = LoadPicture()
```

Project

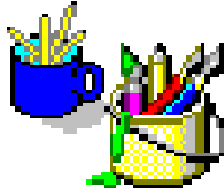


The size of the MyPic.BMP picture is not exactly the same as the size of the Picture control inside Form1. To adjust the size of the Picture control so that the BMP picture will perfectly fit inside the Picture control you can do the following:

-  Set the **AutoSize** property of the **Picture1** Picture control to **True**.
-  Set the **AutoSize** property of the **Picture2** Picture control to **True**.
-  Execute the ClipPic program, and note that now the picture controls have the same size of the **MyPic.BMP** picture.
-  Click the Exit button to terminate the ClipPic program.

Cosmetic Considerations






The BackColor Property of the Picture Control

For cosmetic reason, you may want to set the **BackColor** property of the Picture controls to black as follows:

 Set the **BackColor** property of **Picture1** to black.

 Set the **BackColor** property of **Picture2** to black.

 Execute the ClipPic program, and notice that when the picture control does not have a picture in it, the user sees a black background. This make the program easier to use, because the black picture serves as a visual indication that there is no picture inside the Picture control (see Figure 27.9).

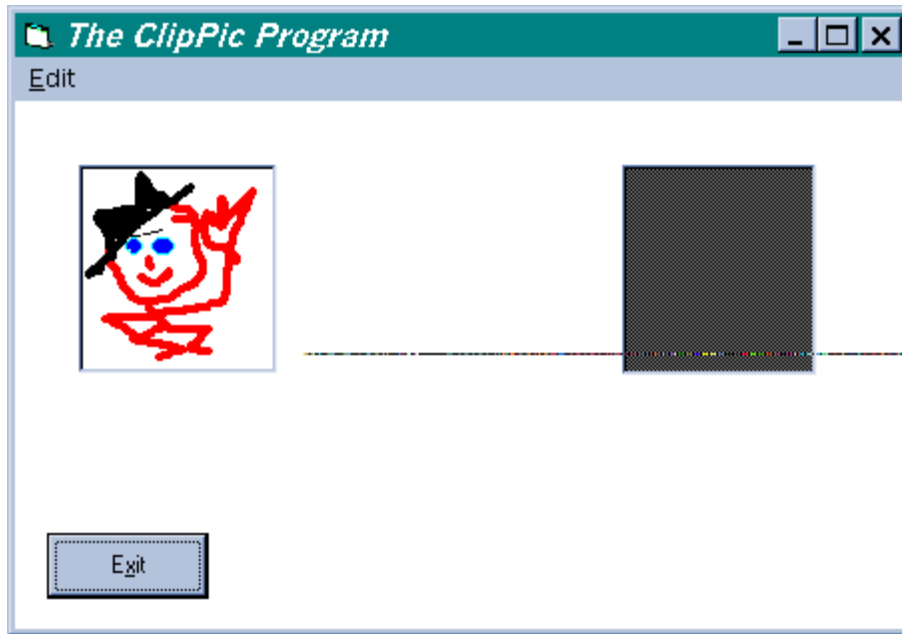



Figure 27.9. When the Picture control does not have a picture in it, it appears black.

 Experiment with the ClipPic program, then click the Exit button to terminate the program.





NOTE

For cosmetic purposes, make sure that the **BorderStyle** property of the **Picture** controls is set to **1-Fixed Single**. This way, the border of the **Picture** control is visible.

The LostFocus and GotFocus Properties

When the **Picture** control is the active control, the ClipPic program does **not** give any visual indication that indeed the Picture control has the focus. You can add code that gives a Visual indication that the Picture control has the focus.

 Place a **Shape** control inside Form1.


 Set the properties of the Shape control as follows:

Name: Shape1


BorderStyle: 1-Solid

FillStyle: 1-Transparent

Shape: 0-Rectangle

 Drag and size the **Shape1** control so that it will surround the **Picture1** control (see Figure 27.10).

 Place another Shape control inside Form1.


 Set the properties of the second Shape control as follows:

Name: Shape2

BorderStyle: 1-Solid

FillStyle: 1-Transparent

Shape: 0-Rectangle

 Drag and size the **Shape2** control so that it will surround the **Picture2** control (see Figure 27.10).

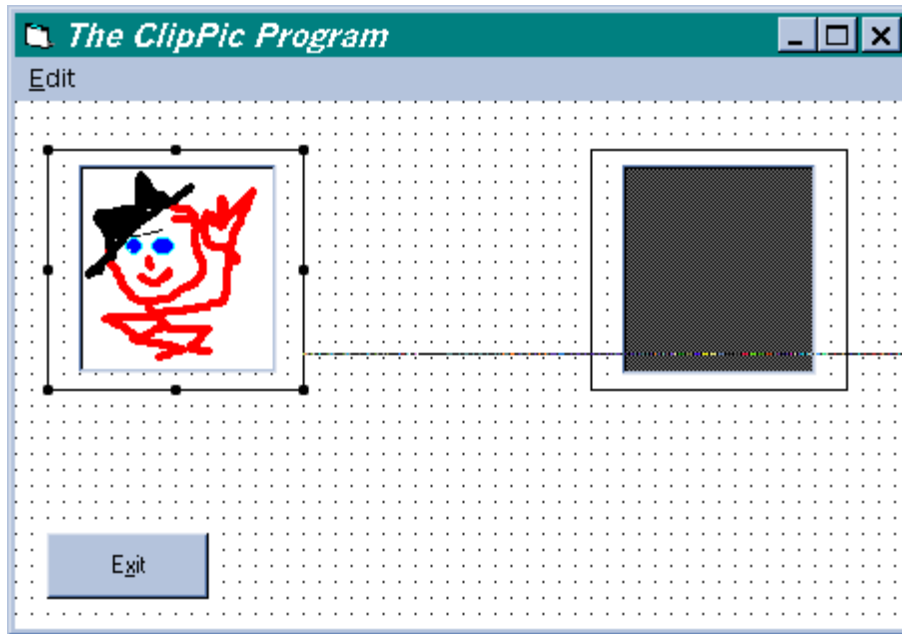



Figure 27.10. Enclosing the Picture controls with Shape controls.


The idea is as follows:

- When the Picture control will have the focus (when the Picture control will be the active control), the sides of the rectangle that surrounds the Picture control will appear as **dashed** lines.
- When the Picture control will **not** have the focus (when the Picture control will not be the active control), the sides of the rectangle that surrounds the Picture control will appear as **solid** lines.

Here is how you implement this:

 Double click the Picture1 control to display the Code window, set the **Object** list box to **Picture1** and set the **Proc** list box to **GotFocus**.

Visual Basic responds by displaying the **Picture1_GotFocus()** procedure ready to be edited by you.


 Type the following code inside the **Picture1_GotFocus()** procedure:

```
Private Sub Picture1_GotFocus()
```


```
Shape1.BorderStyle = 2
```

```
End Sub
```

The **GotFocus** event occurs automatically whenever **Picture1** gets the focus (for example, whenever the user clicks inside **Picture1**). So whenever **Picture1** receives the focus, the **BorderStyle** property of **Shape1** is set to **2** (**dashed** lines).

 Double click the **Picture1** control to display the Code window, set the **Object** list box to **Picture1** and set the **Proc** list box to **LostFocus**.

Visual Basic responds by displaying the **Picture1_LostFocus()** procedure ready to be edited by you.

 Type the following code inside the **Picture1_LostFocus()** procedure:

```
Private Sub Picture1_LostFocus()
```


```
Shape1.BorderStyle = 1
```

```
End Sub
```

The **LostFocus** event occurs when the control loses the focus, For example, if currently **Picture1** has the focus and then the user clicks inside **Picture2**, the **Picture1_LostFocus()** procedure is executed automatically.

The code that you typed inside the **Picture1_Lost_Focus()** procedure sets the **BorderStyle** property of **Shape1** back to **1** (**solid** border).

=====


 Type the following code inside the **Picture2_GotFocus()** procedure:

```
Private Sub Picture2_GotFocus()
```

```
Shape2.BorderStyle = 2
```

```
End Sub
```

The preceding code is automatically execute whenever the Picture2 control gets the focus,. The code that you typed displays the border of the **Shape2** control as **dashed** lines.

 Type the following code inside the **Picture2_LostFocus()** procedure:

```
Private Sub Picture2_LostFocus()
```


```
Shape2.BorderStyle = 1
```

```
End Sub
```

The preceding code is automatically executed when the Picture2 control loses the focus. In this case, the sides of the **Shape2** control are displays as **solid** lines.


Let's see your code in action:

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

 Execute the ClipPic program. Verify that whenever the Picture control has the focus, the sides of the rectangle that surrounds the Picture control are dashed lines. If the Picture control does not have the focus,

=====

the sides of the rectangle that surround the Picture control are solid lines. (See Figure 27.11)

 Experiment with the ClipPic program, then click the Exit button of terminate the program.

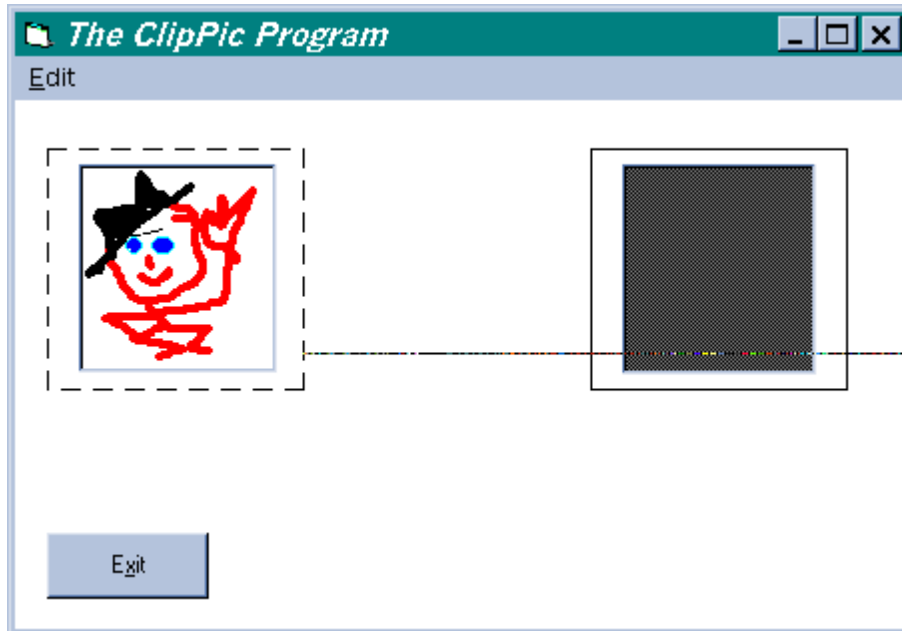


Figure 27.11. A dashed rectangle indicates that the Picture control has the keyboard focus.

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