

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 28 - Loading Pictures During Runtime


In this lesson you'll learn to write programs that load pictures during runtime.


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\Lesson28** 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 **LoadPic.FRM** inside the **C:\VBMyProg\Lesson28** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **LoadPic.VBP** inside the **C:\VBMyProg\Lesson28** 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 LoadPic Program

You'll design the window of the LoadPic program so that it will look as shown in Figure 28.1.

 Set the properties of Form1 as follows:

Name: Form1

Caption: The LoadPic Program

BackColor: White

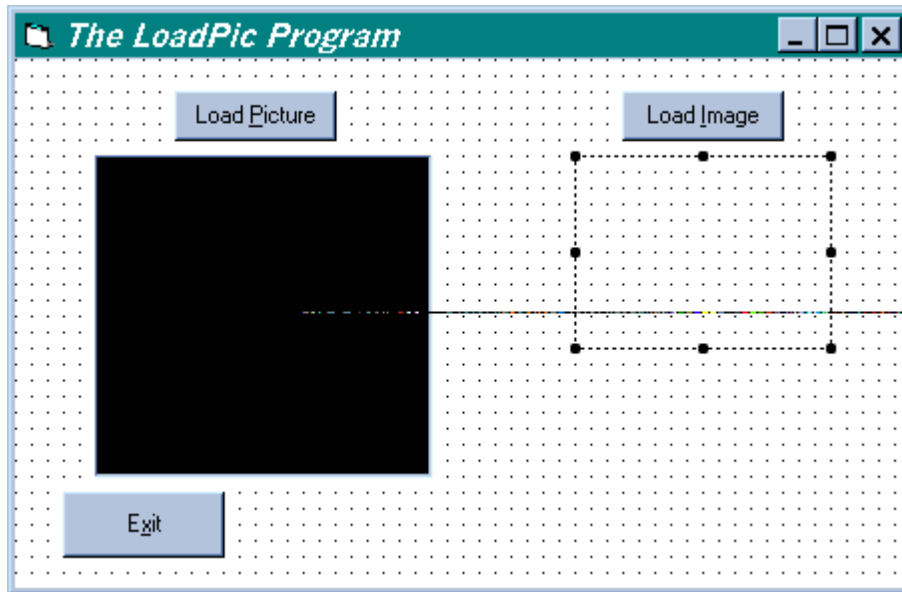



Figure 28.1. Form1 of the ClipPic program

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 LoadPic program terminates itself.

Creating the BMP Directory

The LoadPic program that you'll implement in this lesson use a BMP file. Thus, you'll now create the directory where the BMP file will reside.

 Create the **C:\VBMyProg\Lesson28\BMP** directory.

 Copy the **Pole.BMP** file from the **\TegoVB4\VBMyProg\BMP** directory to the **C:\VBMyProg\Lesson28\BMP** directory that you created in the previous step.

Placing a Picture Control Inside Form1

You'll now place a Picture control inside Form1:


 Place a Picture control inside Form1. Then set the properties of the Picture control as follows:

Name: Picture1

BorderStyle: 1-Fixed Single

BackColor: Black

AutoSize: False

 Drag the picture control to the left and size it as shown in Figure 28.1.

Placing an Image Control Inside Form1

You'll now place an Image control inside Form1:

 Place an Image control inside Form1. Then set the properties of the Image control as follows:

Name: Image1

BorderStyle: 1-Fixed Single

Stretch: False


Placing the Load Picture and Load Image Buttons

You'll now place two CommandButtons inside Form1:

 Place a CommandButton inside Form1. Then set the properties of the CommandButton as follows:

Name: cmdLoadPicture

Caption: Load &Picture

 Place another CommandButton inside Form1. Then set the properties of the CommandButton as follows:


Name: cmdLoadImage

Caption: Load &Image

Loading a Picture

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

=====

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

```
Private Sub cmdLoadPicture_Click()  
  
Dim ProgramPath  
Dim BMPPath  
  
ProgramPath = App.Path  
If Right(ProgramPath, 1) <> "\" Then  
    ProgramPath = ProgramPath + "\"  
End If  
  
BMPPath = ProgramPath + "BMP\  
  
Picture1.Picture = LoadPicture(BMPPath + "Pole.BMP")  
  
End Sub
```

You declared two local variables:

```
Dim ProgramPath  
Dim BMPPath
```

You then set the value of the **ProgramPath** variable to **App.Path**:

```
ProgramPath = App.Path
```

App.Path is automatically maintained by Visual Basic. **App.Path** is a string that contains the directory from which the program is executed.

As it turns out, **App.Path** may or may not attach a backslash character (\) as the last character of the string that represents the directory from which the program

=====

is executed. For example, if the program is executed from the **C:\VBMyProg\Lesson28** directory, then **App.Path** is equal to **C:\VBMyProg\Lesson28**. However, if the program resides inside the root directory **C:** then **App.Path** returns **C:** (not **C:**). So you execute an **If** statement that checks if the rightmost character is equal to: ****. If the rightmost character is not the backslash character, the **If** condition is satisfied, and the code under the **If** statement adds a **** character to the string:

```
If Right(ProgramPath, 1) <> "\" Then
    ProgramPath = ProgramPath + "\"
End If
```

Note that the **Right()** function is used. **Right(ProgramPath,1)** returns the rightmost character of the **ProgramPath** string. For example, if **ProgramPath** is equal to **C:\VBMyProg\Lesson28**, then **Right(ProgamPath,1)** returns the **8** character (which is the rightmost character).

To summarize, the variable **ProgramPath** now contains the path of the directory from which the program is executed, and the last character is the **** character.

You then set the value of the **BMPPath** variable:

```
BMPPath = ProgramPath + "BMP\"
```

For example, if **ProgramPath** is equal to **C:\VBMyProg\Lesson28**, then **BMPPath** is set to:

```
C:\VBMyProg\Lesson28\ + BMP\ = C:\VBMyProg\Lesson28\BMP\
```

Finally, the **LoadPicture()** function is executed:

```
Picture1.Picture = LoadPicture(BMPPath + "Pole.BMP")
```

So if the program is executed from the **C:\VBMyProg\Lesson28** directory, the string that you supplied to the **LoadPicture()** function is:

=====

C:\VbMyProg\Lesson28\BMP\Pole.BMP

Let's see your code in action:

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

 Execute the LoadPic program.

The window of the LoadPic program appears as shown in Figure 28.2.

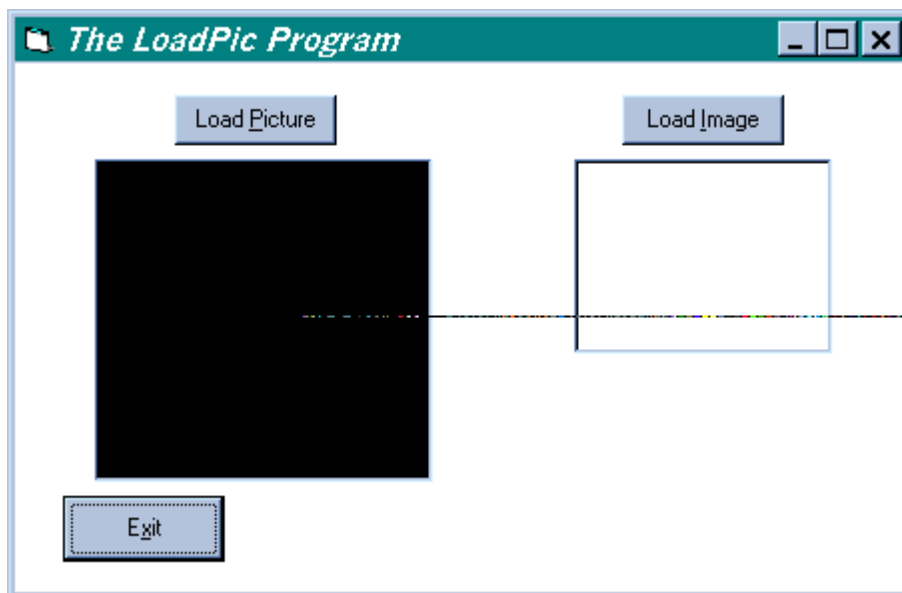



Figure 28.2. *The window of the LoadPic program. (No picture or image was loaded yet).*

 Click the **Load Picture** button, and verify that the picture control displays the **Pole.BMP** picture as shown in Figure 28.3.

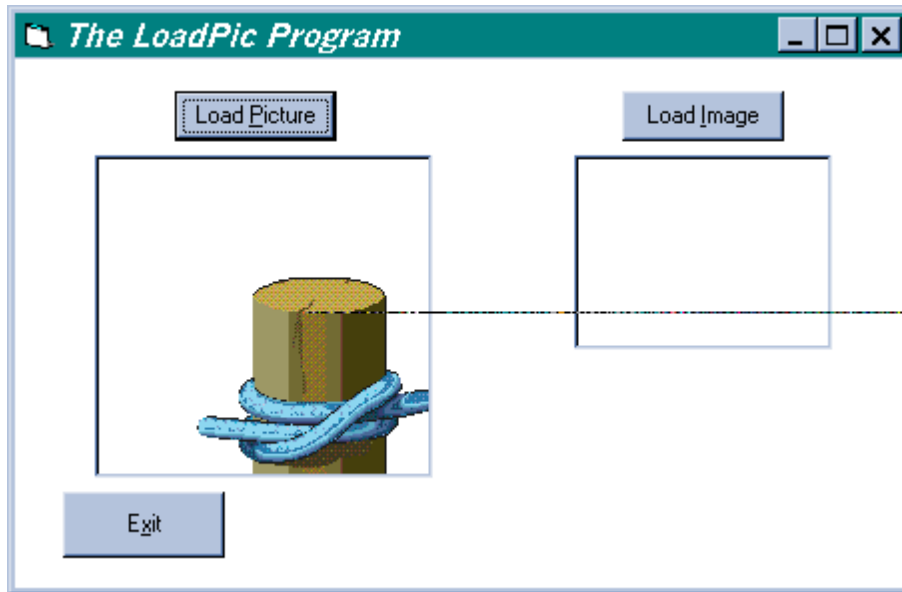


Figure 28.3. Loading the Pole.BMP picture.

If you load the **Pole.BMP** picture with Paint or Paintbrush, you'll see the **Pole.BMP** picture as shown in Figure 28.4.

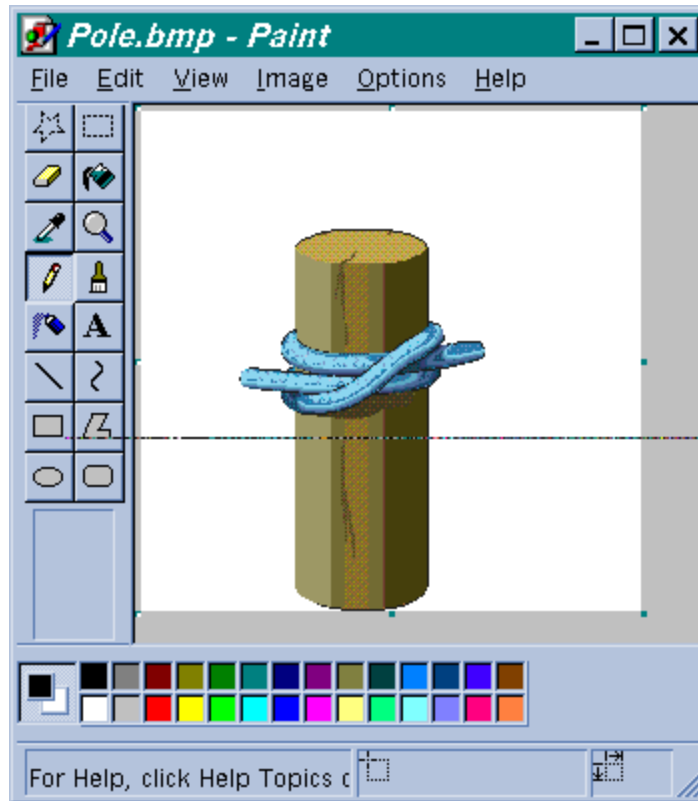





Figure 28.4. *The Pole.BMP picture.*

The reason you do not see the entire **Pole.BMP** picture inside the Picture control, is that the size of the Picture control is too small to hold the entire **Pole.BMP** picture. You could set the **AutoSize** property of the Picture control to **True**, and this will stretch the Picture control so that the entire Pole.BMP picture will fit inside the Picture control. However, in many cases, you do not want to make the Picture control larger than the size that you set for it during design time, because this will mess up the entire form. To see this in action, perform the following experiment:

-  Click the Exit button to terminate the LoadPic program.

-  Set the **AutoSize** property of Picture1 to **True**.

-  Select **Save Project** from the **File** menu of Visual Basic to save your work. Then execute the LoadPic program.

 Click the **Load Picture** button.

The LoadPic program loads the Pole.BMP picture. Because you set the **AutoSize** property of the Picture control to **True**, the Picture control stretched itself as shown in Figure 28.5. As you can see, this messes up the form.

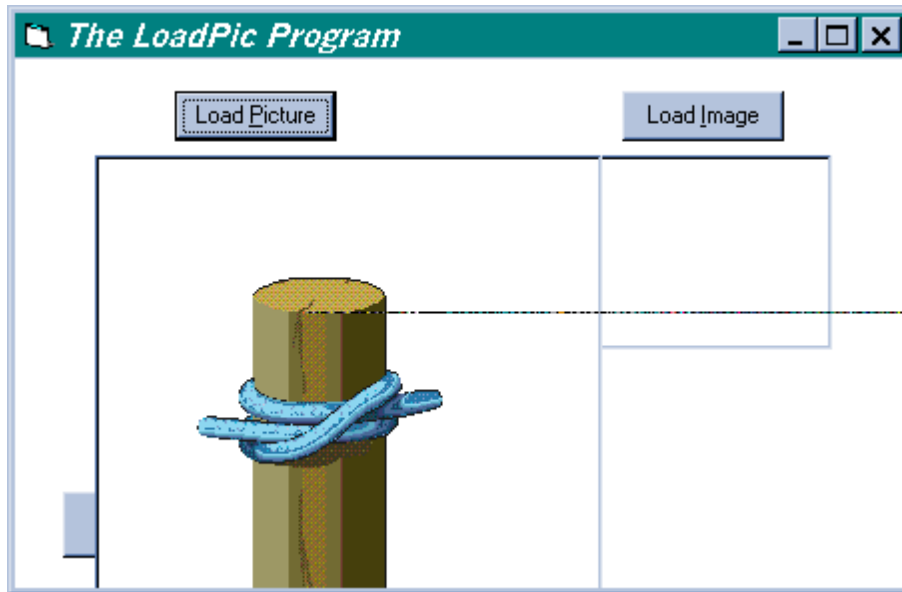




Figure 28.5. *The Picture control was stretched itself too much.*

 Click the Exit button to terminate the LoadPic program.

 Set the **AutoSize** property of Picture1 back to **False**.

Loading an Image

You'll now attach code to the **Click** event of the **cmdLoadImage** button.

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

```

Private Sub cmdLoadImage_Click()

Dim ProgramPath
Dim BMPPath

ProgramPath = App.Path
If Right(ProgramPath, 1) <> "\" Then
    ProgramPath = ProgramPath + "\"
End If

BMPPath = ProgramPath + "BMP\"

Image1.Picture = LoadPicture(BMPPath + "Pole.BMP")

End Sub

```

The code that you typed inside the **cmdLoadImage_Click()** procedure is almost identical to the code that you typed inside the **cmdLoadPicture_Click()** procedure. In fact, the only statement that is different is the statement that loads the BMP picture:

```
Image1.Picture = LoadPicture(BMPPath + "Pole.BMP")
```

So now you use the **LoadPicture()** function to load a picture into the Image control.

Let's see your code in action:

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

 Execute the LoadPic program.

 Click the **Load Image** button.

The Pole.BMP picture is loaded into the Image control as shown in Figure 28.6.

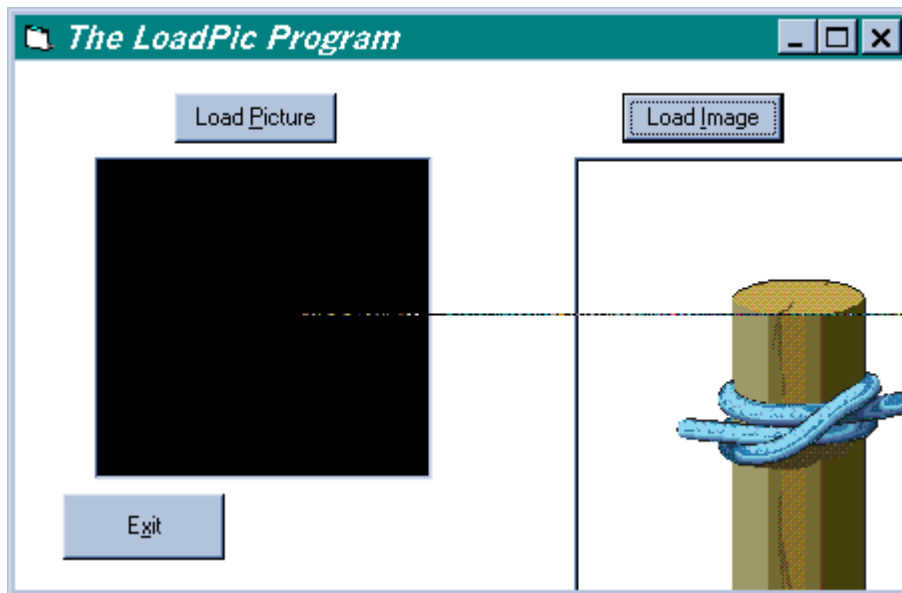




Figure 28.6. Loading the Pole.BMP picture into the Image control.

As you can see from Figure 28.6, once you load the Pole.BMP picture, the Image control stretches itself so that the entire Pole.BMP picture fits inside the Image control. This messes up the form. However, this can be easily fixed as follows:

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

 Set the **Stretch** property of the Image1 image control to **True**.

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

 Execute the LoadPic program.

 Click the **Load Image** button.

The Pole.BMP picture is loaded into the Image control as shown in Figure 28.7. (The Pole.BMP picture shrunk itself so that it will fit inside the Image control)

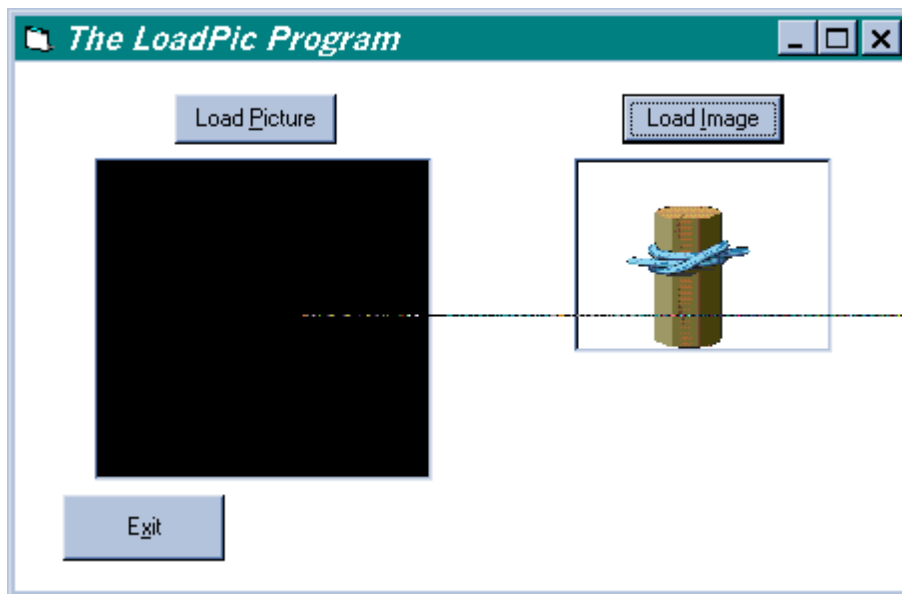



Figure 28.7. *The Pole.BMP picture loaded into the Image control. The Stretch property of the Image control is set to True.*

 Click the Exit button to terminate the program.

The thing to note about the LoadPic program is that you use the **LoadPicture()** function to load the pictures during runtime. You use the **LoadPicture()** function to load a picture into the Picture control as well as into the Image control.

As you saw , you can set the **AutoSize** property of the Picture control to **True**, and this will size the Picture control so that its area will fit the area of the loaded picture. However, this may mess up your form. If the **AutoSize** property of the Picture control is set to **False**, the size of the Picture control will be the same

size as you set the Picture control during design time, but the entire picture may not fit inside the control.

And when loading pictures into the Image control, if you set the **Stretch** property to **False**, the Image control will stretch itself so that the Image control has the original size of the loaded picture (but this may mess up the form). If you set the **Stretch** property of the Image control to **True**, the loaded picture will stretch/shrink itself to fit inside the Image control.

What You Accomplished in This Lesson



You completed Lesson 28 of the Self Study Visual Basic tutorial. In this lesson you learned to write programs that load pictures during runtime. You learned to load picture into the Picture and Image controls. You also learned about the **AutoSize** property of the Picture control and about the **Stretch** property of the Image control.

Frequently Asked Questions



Q1. Should I load pictures during runtime, or should I set the **Picture** property during design time?

A1. It depends on the particular application that you are developing. Sometimes, your application lets your user select the BMP file during runtime. Naturally, in this case you can't set the **Picture** property during design time.

it is important to understand that if you set the **Picture** property of a Picture control or an Image control during design time, then the BMP picture becomes an integral part of the application. So when distributing the application, you do not have to also distribute the BMP files.

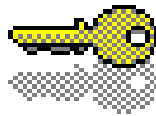
=====

Exam



1. Let's say that your application loaded a picture into a Picture control or into an Image control during runtime using the **LoadPicture()** function. Then, during the execution of your program, you want to delete the picture so that the Picture (or Image control) will not have any picture in it. What function will you use?

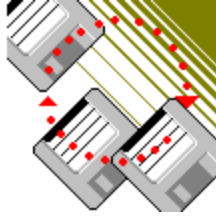
Answers to Exam




1. To delete a picture from a Picture control or from an Image control during runtime, use the **LoadPicture()** function without specifying any value for the parameter of the **LoadPicture()** function. For example, the following statement sets the **Picture** property of the `imgMyImage` control to null:

```
imgMyImage.Picture=LoadPicture()
```

Project



The **Stretch** property of the Image control enables you to perform various image effects. Here is an example:

 Type the following code inside the **Image1_Click()** procedure of the LoadPic program:

```
Private Sub Image1_Click()  
  
Image1.Height = Image1.Height + 100  
Image1.Width = Image1.Width + 100  
  
If Image1.Height >= 3000 Then  
  
    Image1.Height = 1455  
    Image1.Width = 1935  
  
End If  
  
End Sub
```

The code that you typed inside the **Image1_Click()** procedure is executed whenever the user clicks the **Image1** control.

The code inside the **Image1_Click()** procedure increases the **Height** property of the **Image1** image control by **100**, and it increases the **Width** property of the **Image1** image control by **100**:

```
Image1.Height = Image1.Height + 100
```

=====

```
Image1.Width = Image1.Width + 100
```

An **If** statement is then executed to check that the **Height** property does not exceed 3000:

```
If Image1.Height >= 3000 Then
```

```
    Image1.Height = 1455
```

```
    Image1.Width = 1935
```

```
End If
```

If the **Height** property exceeds 3000, the **Height** and **Width** properties are set back to their original setting (the setting that the Image control has during design time).


Let's see your code in action:


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

 Execute the LoadPic program.

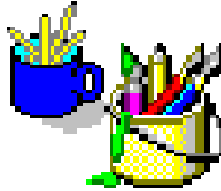
 Click the **Load Image** button.

The LoadPic program responds by loading the Pole.BMP picture into the Image control.


 Click the **Image1** control a few times, and note that every click causes the Image1 control and its picture to increase in size. When the Image1 control reaches a certain size, the original size is restore.

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


Cosmetic Considerations



In the Project section you added code that increases the size of the Image control every time the user clicks the Image control. Some cosmetically pleasant effects can be easily achieved. For example, instead of having to click the Image control, you can perform the clicking automatically as follows:

 Set the **BorderStyle** property of the **Image1** image control to **0-None**.

 Place a **Timer** control inside Form1.

 Set the properties of the **Timer** control as follows:

Name: Timer1

Enabled: True

Interval: 50

 Type the following code inside the **Timer1_Timer()** procedure:

```
Private Sub Timer1_Timer()
```

```
Image1_Click
```

```
End Sub
```


The **Timer1_Timer()** procedure is executed automatically every **50** milliseconds (because you set the **Interval** property to **50**, and you set the **Enabled** property to **True**). The code that you typed inside the **Timer1_Timer()** procedure executed the **Image1_Click()** procedure.

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

 Execute the LoadPic program.

 Click the **Load Image** button.

As you can see, the picture inside the Image1 control stretches continuously.

 Click the Exit button to terminate the LoadPic program.

Other Image Effects

You saw that you can perform image effects by setting the **Stretch** property of the Image control to **True** and then dynamically setting the **Height** and **Width** properties of the Image control.

Likewise, you can change the **Left** and **Top** properties of the Image control during runtime, and this causes the picture to move inside the window of the application.

Of course, you can change all four properties of the Image control (**Top**, **Left**, **Height**, **Width**), and this will result an image effect of an image that dynamically stretched/shrunked and moves inside the window of the application.



Typically, when moving images, you set the **BorderStyle** property of the Image control to **0-None** (so that the Image control will not a frame around it).

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