

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 16 - Drawing Lines (Line Control)


In this lesson you'll learn to draw lines using the Line control.


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\Lesson16** 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 **MyLine.FRM** inside the **C:\VBMyProg\Lesson16** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **MyLine.VBP** inside the **C:\VBMyProg\Lesson16** directory.

Always Use Option Explicit

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

Option Explicit

Placing the Line Control

You'll now place the Line control inside Form1. The icon of the Line control inside the Toolbox of Visual Basic is shown in Figure 16.1.

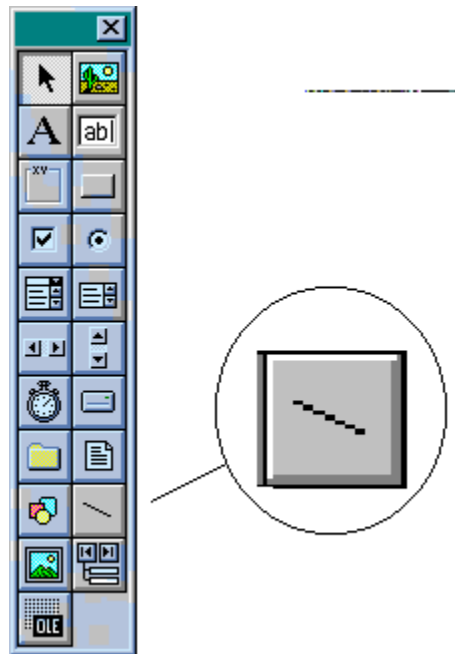



Figure 16.1. *The icon of the Line control inside the Toolbox.*

NOTE

In Figure 16.1 the icon of the Line control is shown as the icon on the right column and third row from the bottom.

 Make sure Form1 is the selected window, and then double click the Line control inside the Toolbox.

Visual Basic responds by placing the Line control inside Form1 (see Figure 16.2).

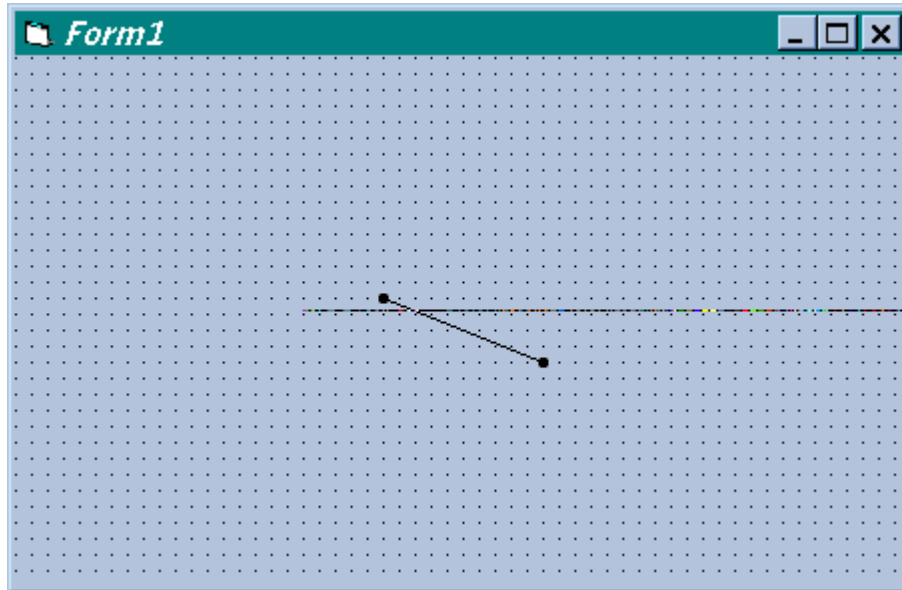




Figure 16.2. *Placing the Line control inside Form1.*

 When the Line control is the selected object, you see two small black squares on its endpoints (for example, in Figure 16.2, the Line is shown selected). To set the properties of the Line control that you placed inside Form1, make the Line the selected object, and then press **F4** on your keyboard to display the Properties window of the Line control. Set the properties of the Line control as follows:

Name: linMyLine

Placing the Move Line Button Inside Form1

You'll now place a CommandButton inside Form1. This button will serve to move the **linMyLine** line inside Form1.


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

Name: cmdMoveLine

Caption: &Move Line

Placing an Exit Button

You'll now place an Exit button inside Form1.

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

Name: cmdExit

Caption: E&xit

Form1 should now look as shown in Figure 16.3.

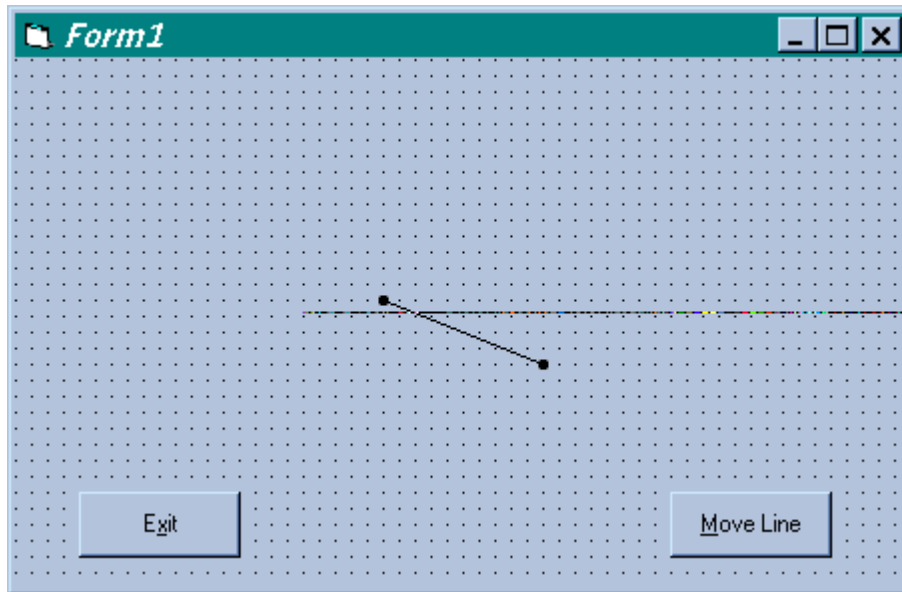


Figure 16.3. *Form1 with the Line control, Move Line button, and Exit button in it.*

Attaching Code to the Exit Button

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

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

```
Private Sub cmdExit_Click()
```

```
End
```


```
End Sub
```

Before adding any additional code to the MyLine program, let's see what you accomplished so far:

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

 Execute the MyLine program.

The window of the MyLine program appears as shown in Figure 16.4. As you can see in Figure 16.4, the Line control appears as a straight line inside the window of the program.

 Terminate the MyLine program.

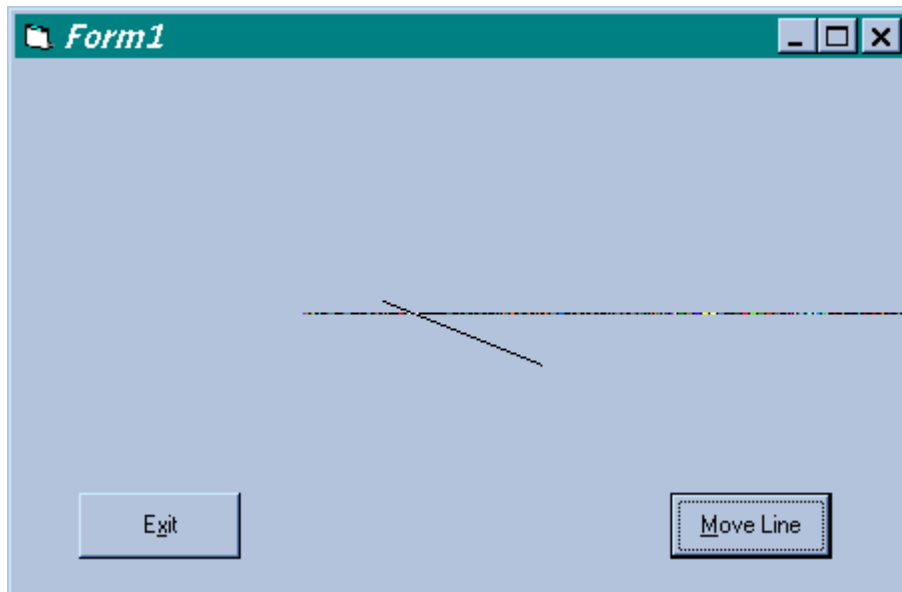



Figure 16.4. *The MyLine program.*

Attaching Code to the Move Line Button

You'll now attach code to the **Click** event of the **Move Line** button. In particular, you'll attach code that moves the line to a random location.

 Attach the following code inside the `cmdMoveLine_Click()` procedure:

```
Private Sub cmdMoveLine_Click()
```

=====

```
Randomize
linMyLine.X1 = Form1.ScaleWidth * Rnd()
linMyLine.X2 = Form1.ScaleWidth * Rnd()
linMyLine.Y1 = Form1.ScaleHeight * Rnd()
linMyLine.Y2 = Form1.ScaleHeight * Rnd()
```

End Sub

Let's go over the code that you typed inside the **cmdMoveLine_Click()** procedure.

You initialize the random number generator:

```
Randomize
```

You then typed the following statement:

```
linMyLine.X1 = Form1.ScaleWidth * Rnd()
```

ScaleWidth is a property of Form1 that indicates the width of Form1. You multiply **Form1.ScaleWidth** by **Rnd()**. Because **Rnd()** returns a random number equal to 0 or greater than 0 and less than 1, the result of the multiplication is a number equal or greater to 0 and less than **Form1.ScaleWidth**.

X1 is a property of the Line control. The line has two endpoints. One endpoint has the coordinates **X1,Y1**. **X1** is the distance between the endpoint of the line and the left edge of the Form1. **Y1** is the distance between the endpoint of the line and the top edge of Form1.

The other endpoint of the line has the coordinates **X2,Y2**. These coordinates represent the location of the endpoint (see Figure 16.5).

=====

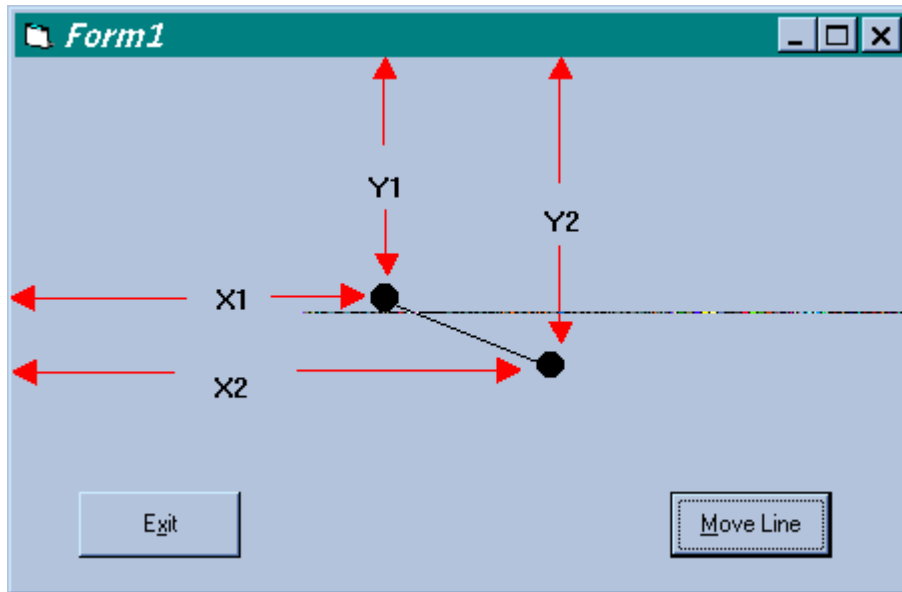


Figure 6.5. The X1, X2, Y1 and Y2 properties of the Line control.

You set the X2, Y1, and Y2 properties of the linMyLine control as follows:

```
linMyLine.X2 = Form1.ScaleWidth * Rnd()
linMyLine.Y1 = Form1.ScaleHeight * Rnd()
linMyLine.Y2 = Form1.ScaleHeight * Rnd()
```

So putting it all together, whenever the user clicks the **Move Line** button, the coordinates of the line are changed to a random location.



NOTE

The **ScaleWidth** property of a form represents the width of the client area of the form.

The **ScaleHeight** property of a form represents the width of the client area of the form.

Let's see you code in action:





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

 Execute the MyLine program.

 Click the **Move Line** button.

The MyLine program responds by displaying the line at a random location.

 Keep clicking the **Move Line** button. Because the **M** in **Move Line** is underlined, you can press **Alt+M** on your keyboard to continuously move the line to new locations.

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

What You Accomplished in This Lesson



You completed Lesson 16 of the Self Study Visual Basic tutorial. In this lesson you learned to draw lines using the Line control.

Frequently Asked Questions



Q1. Can I draw thicker lines?

A2. Yes, see the Project section of this lesson.

Q1. Can I draw the lines using colors?

A2. Yes, see the Cosmetic Considerations section of this lesson.

Q2. Can I use more than one Line control in the same form?

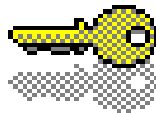
A3. Yes, just as you can place more than one CommandButton inside the form, you can place more than one Line control inside the see the form.

Exam



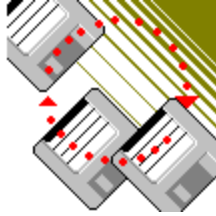
1. The **ScaleWidth** property of the form represents the _____
2. The **ScaleHeight** property of the form represents the _____

Answers to Exam



1. The **ScaleWidth** property of the form represents the width of the form.
The **ScaleHeight** property of the form represents the height of the form.

Project



Modify the MyLine program so that the width of the line changes randomly.

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

```
Private Sub cmdMoveLine_Click()  
Dim MaxNum  
Dim MinNum  
  
Randomize  
  
linMyLine.X1 = Form1.ScaleWidth * Rnd()  
linMyLine.X2 = Form1.ScaleWidth * Rnd()  
linMyLine.Y1 = Form1.ScaleHeight * Rnd()  
linMyLine.Y2 = Form1.ScaleHeight * Rnd()  
  
MaxNum = 50  
MinNum = 1  
linMyLine.BorderWidth = _  
    Int((MaxNum - MinNum + 1) * Rnd() + MinNum)  
  
End Sub
```

You added the declarations of the following variables:

```
Dim MaxNum  
Dim MinNum
```

At the end of the procedure you added the following statements:

```
MaxNum = 50
MinNum = 1
linMyLine.BorderWidth = _
    Int((MaxNum - MinNum + 1) * Rnd() + MinNum)
```

The preceding statements set the **BorderWidth** property of the linMyLine control to a random integer where the integer can be between **1** and **50**.



NOTE

The **BorderWidth** property of the Line control determines the thickness of the line. For example when the **BorderWidth** property of the line is equal to **1**, the width of the line is **1** pixel. When the **BorderWidth** property of the line is equal to **10**, the width of the line is **10** pixels, and so on.



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

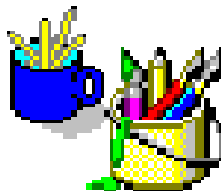


Click the **Move Line** button several times, and note that the width of the line changes randomly after each click.



Experiment with the MyLine program. And then click the **Exit** button to terminate the program.

Cosmetic Considerations



When designing programs that utilize the Line control, you can make your programs more attractive looking by drawing the lines in various colors.

Modify the MyLine program so that the lines are drawn in random colors:

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

```
Private Sub cmdMoveLine_Click()  
Dim MaxNum  
Dim MinNum  
  
Randomize  
  
linMyLine.X1 = Form1.ScaleWidth * Rnd()  
linMyLine.X2 = Form1.ScaleWidth * Rnd()  
linMyLine.Y1 = Form1.ScaleHeight * Rnd()  
linMyLine.Y2 = Form1.ScaleHeight * Rnd()  
  
MaxNum = 50  
MinNum = 1  
linMyLine.BorderWidth = _  
    Int((MaxNum - MinNum + 1) * Rnd() + MinNum)  
  
MaxNum = 15  
MinNum = 0  
linMyLine.BorderColor = _  
    QBColor(Int((MaxNum - MinNum + 1) * Rnd() + MinNum))  
  
End Sub
```

The code that you added sets the **BorderColor** property of the **linMyLine** control as follows:

```
MaxNum = 15
MinNum = 0
linMyLine.BorderColor = _
    QBColor(Int((MaxNum - MinNum + 1) * Rnd() + MinNum))
```

The preceding statements generate a random integer between **0** and **15**. This random integer is supplied as the parameter of the **QBColor()** function, and the returned value of the **QBColor()** function is assigned to the **BorderColor** property of the **linMyLine** line.



NOTE

The **BorderColor** property of the Line control determines the color of the line. For example when the **BorderColor** property of the line is equal to **0**, the color of the line is black. When the **BorderColor** property of the line is equal to **15**, the color of the line is white, and so on.



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



Click the **Move Line** button several times, and note that the color of the line changes randomly after each click.



Experiment with the MyLine program. And 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

