

**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](mailto:tegosoft@msn.com)

# Lesson 18 -

## Drawing Lines and Rectangles With the Line Method


In a previous lesson you learned to draw lines using the Line control. In this lesson you'll learn to design programs that draw lines. However, you'll draw the lines using the **Line** method.


### **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\Lesson18** 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 **Lines.FRM** inside the **C:\VBMyProg\Lesson18** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **Lines.VBP** inside the **C:\VBMyProg\Lesson18** 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.

## Setting the Properties of Form1

You'll now design Form1 as shown in Figure 18.1.

 Set the properties of Form1 as follows:


**Name:** Form1

**Caption:** The Lines Program

**BackColor:** White

## Placing the Exit Button

You'll now place an Exit button inside Form1.

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

**Name:** cmdExit

**Caption:** E&xit

 Type the following code inside the **cmdExit\_Click()** procedure.


```
Private Sub cmdExit_Click()
```

```
End
```

```
End Sub
```

## Placing a Timer Control

You'll now place a Timer control inside Form1.

 Place a Timer control inside Form1, and then set the properties of the Timer control that you placed as follows:

**Name:** Timer 1

**Enabled:** True

**Interval:** 10

Your Form1 should now look as shown in Figure 18.1.

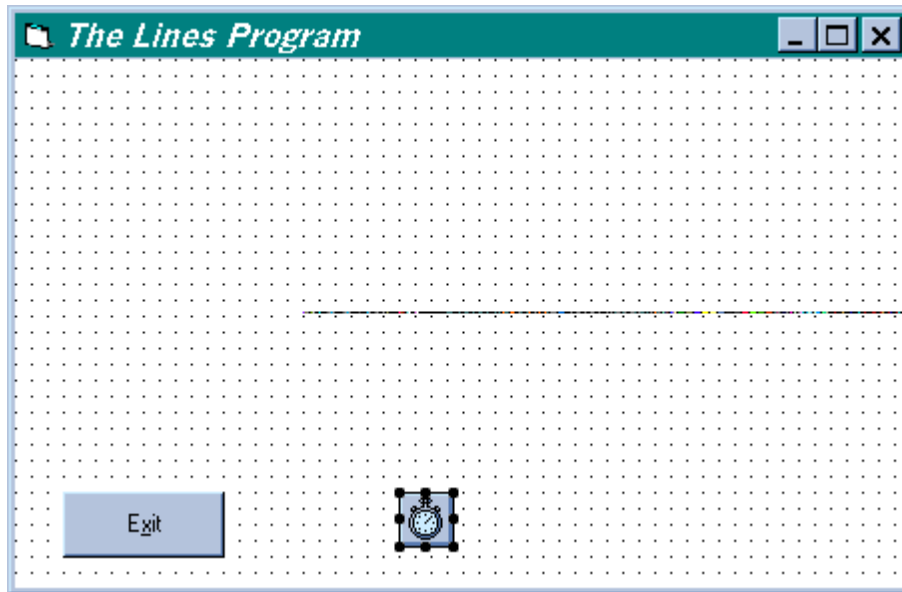


Figure 18.1. Form1 during design time.

## Attaching Code to the Timer Event of the Timer Control

You'll now attach code to the `Timer1_Timer()` procedure:

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

```
Private Sub Timer1_Timer()  
  
Dim x, y  
  
x = Form1.ScaleWidth * Rnd()  
y = Form1.ScaleHeight * Rnd()  
  
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    RGB(255, 0, 0)
```

**End Sub**

The code that you typed declares the **x** and **y** variables:

```
Dim x, y
```

You then assigned random numbers to the **x** and **y** variables:

```
x = Form1.ScaleWidth * Rnd()  
y = Form1.ScaleHeight * Rnd()
```

**x** is assigned with a number that is equal to 0 or greater than 0 but less than the width of the form. **y** is assigned with a number that is equal to 0 or greater than 0 but less than the height of the form.

Finally, you draw a line by executing the **Line** method as follows:

```
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    RGB(255, 0, 0)
```

Let's take a look at the parameters of the **Line** method.

One endpoint of the line is specified as follows:

```
(Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)
```

**(Form1.ScaleWidth/2, Form1.ScaleHeight/2)** represent the coordinates of the center point of Form1. So one endpoint of the line is at the center of Form1.

Then you typed the character: -

=====

After the character - you typed the following coordinates:

(x, y)

After the (x,y) coordinates, you specified the red color as follows:

RGB(255, 0, 0)

Putting it all together, a red line will be drawn from the center of Form1 to a random location inside Form1.

Let's see your code in action:

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

 Execute the Lines program.

As you can see (see Figure 18.2), red lines are drawn from the center of the window to random locations inside the window.

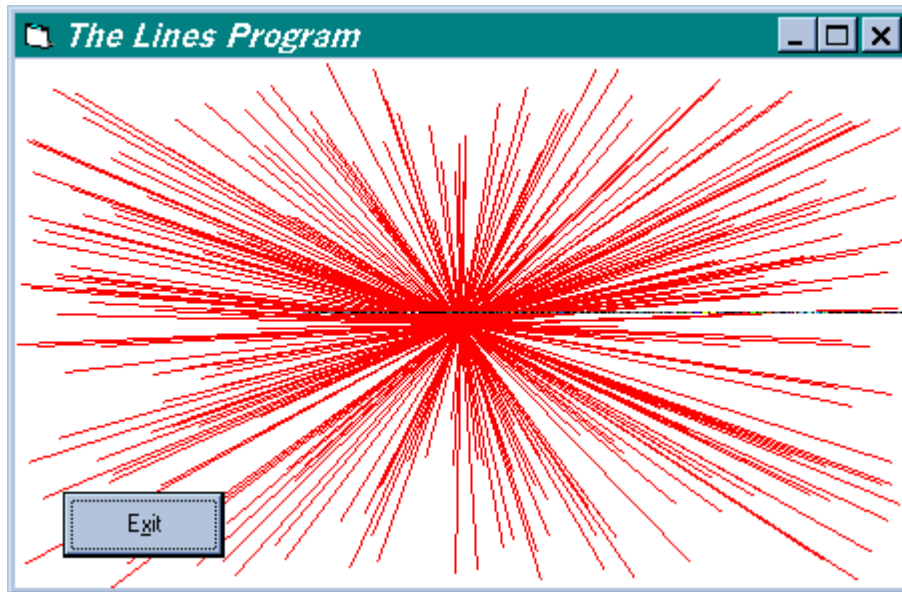



Figure 18.2. *The Lines program in action.*

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

## Drawing Rectangles with the Line Method

You can use the **Line** method to draw rectangles. To draw a rectangle with the **Line** method, you have to add the **B** parameter after the color parameter as follows:

```
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    RGB(255, 0, 0),  
    B
```

As you know, when you use the **Line** method, you specify the two endpoints of the line that you want to draw. When you add the **B** parameter, a rectangle will be drawn instead of a line. The two endpoints will serve as the two opposite corners of a rectangle. Let's see this in action:

=====

 Modify the **Timer1\_Timer()** procedure so that it will look as follows:

```
Private Sub Timer1_Timer()  
  
Dim x, y  
Dim R, G, B  
  
x = Form1.ScaleWidth * Rnd()  
y = Form1.ScaleHeight * Rnd()  
  
R = Int(256 * Rnd())  
G = Int(256 * Rnd())  
B = Int(256 * Rnd())  
  
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    RGB(R, G, B), _  
    B  
  
End Sub
```

Let's go over the code that now resides inside the **Timer1\_Timer()** procedure:

The following variables are declared:

```
Dim x, y  
Dim R, G, B
```

**x** and **y** are assigned with values that represent a random location inside Form1:

=====

```
x = Form1.ScaleWidth * Rnd()  
y = Form1.ScaleHeight * Rnd()
```

You assigned random integers in the range 0 to 255 to the **R**, **G**, and **B** variables:


```
R = Int(256 * Rnd())  
G = Int(256 * Rnd())  
B = Int(256 * Rnd())
```

Finally, you draw a rectangle as follows:

```
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    RGB(R, G, B), _  
B
```

The rectangle is drawn so that one of its corners is at the center of Form1, and the other corner is at random location inside Form1. The color that is used to draw the rectangles is a random color.

Let's see your code in action:

 Execute the Lines program.

The Lines program responds by drawing rectangles as shown in Figure 18.3. As you can see, all the rectangles have one corner in the center of the window, and the opposite corner of the rectangle is at random location. The color used to draw the rectangles varies (random colors are used).



**A1.** Yes, you can. Use whichever method is easier for you.

**Q2.** I want to draw filled rectangles (the inside of the rectangle is filled with a color). Can I use the **Line** method to accomplish this?

**A2.** Yes, you can. See the Project section of this lesson.

**Q3.** I can use the Line control to draw a line, and I can use the **Line** method to draw a line. Which way of implementing a line is better?

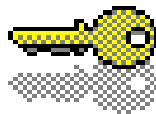
**A3.** If you need to draw a single line (or just a few lines), the Line control can be used. However, if you need to draw many lines (such as the case if your program draws a bar graph that is composed of many lines), then using the Line control is impractical, because you'll need a lot of Line controls. In this case use the **Line** method.

## Exam



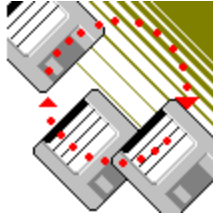
1. To draw a semi circle with the **Line** method you have to: \_\_\_\_\_

## Answers to Exam



1. (This was a trick question) You can only draw straight lines with the **Line** method. You cannot draw a semi circle with the **Line** method.

# Project



Modify the Lines program so that the rectangles will be drawn as filled rectangles (rectangles that are filled with colors).

 Modify the **Timer1\_Timer()** procedure so that it will look as follows:

```
Private Sub Timer1_Timer()
```

```
Dim x, y
```

```
Dim R, G, B
```

```
x = Form1.ScaleWidth * Rnd()
```

```
y = Form1.ScaleHeight * Rnd()
```

```
R = Int(256 * Rnd())
```

```
G = Int(256 * Rnd())
```

```
B = Int(256 * Rnd())
```

```
Form1.Line _
```

```
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _
```

```
    (x, y), _
```

```
    RGB(R, G, B), _
```

```
    BF
```

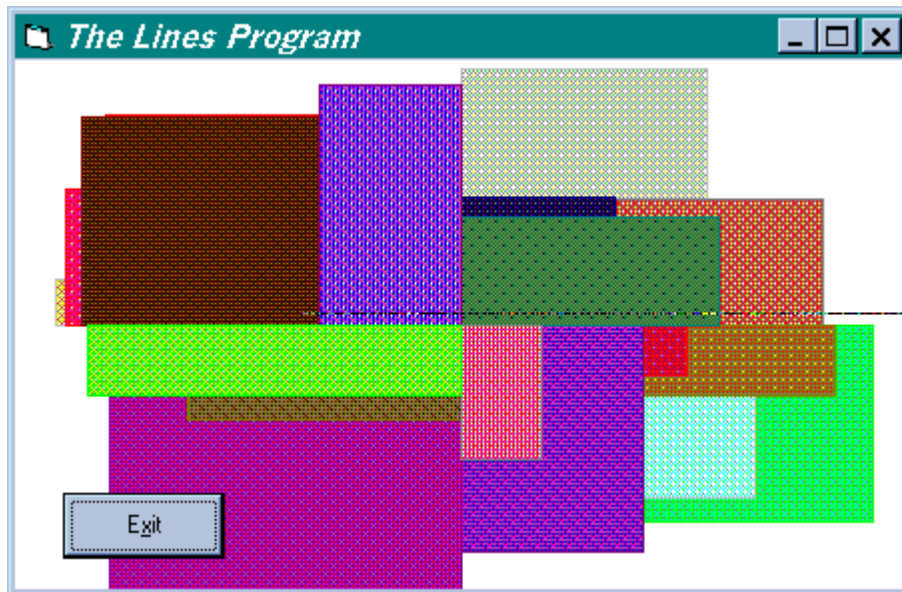
```
End Sub
```

=====

The only modifications that you made to the **Timer1\_Timer()** procedure is that the last parameter of the **Line** method is now **BF**:


```
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    RGB(R, G, B), _  
    BF
```


So now the rectangles are drawn as shown in Figure 18.4 (the rectangles are filled with the color specified as the color parameter of the **Line** method).



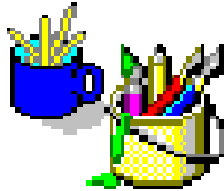
**Figure 18.4.** *The Line program draws filled rectangles.*

Let's see your code in action:

 Execute the Lines program, and verify that the rectangles are filled as shown in Figure 18.4.

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

## Cosmetic Considerations



Take a look at Figure 18.4 which shows the filled rectangles. The rectangles are **not** filled with pretty colors. Rather, the colors look as if there is something wrong with the program! Indeed, there is something wrong.

The statement that draws the filled rectangles inside the **Timer1\_Timer()** procedure is as follows:

```
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    RGB(R, G, B), _  
    BF
```

That is, you assigned the color **RGB(R,G,B)** where **R**, **G**, and **B** can each have an integer value between 0 and 255. As you can see, **RGB()** can represent millions of colors. However, if your video card can only display 256 colors, many of the colors that the **RGB()** function produces will not be displayed as intended by the **RGB()** function. Rather, a distorted color will be produced. Figure 18.4 was captured while setting the system as a 256 colors system. Thus many of the colors are displayed distorted. If the Lines program was executed on a system that is capable of displaying millions of colors, the rectangles would appear with beautiful colors.

If you want to "play it safe" so that all your user will see non distorted colors, you have to modify the **Timer1\_Timer()** procedure as follows:

=====

 Modify the code inside the **Timer1\_Timer()** procedure so that it will look as follows:

```
Private Sub Timer1_Timer()  
  
Dim x, y  
Dim Color  
  
x = Form1.ScaleWidth * Rnd()  
y = Form1.ScaleHeight * Rnd()  
  
Color = Int(16 * Rnd())  
  
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    QBColor(Color), _  
    BF  
  
End Sub
```

You declared the **Color** variable:

```
Dim Color
```

You generated a random integer between **0** and **15**:


```
Color = Int(16 * Rnd())
```


And finally, you display the filled rectangle as follows:

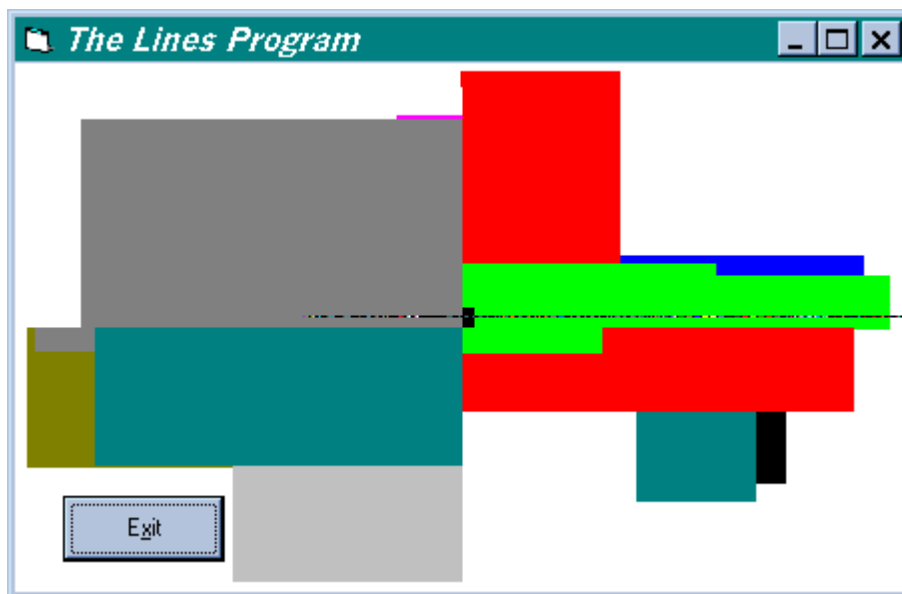
=====

```
Form1.Line _  
    (Form1.ScaleWidth / 2, Form1.ScaleHeight / 2)- _  
    (x, y), _  
    QBColor(Color), _  
    BF
```

So now the rectangle is filled with a color that could be one of 16 possible colors (not one of millions of possible colors).

 Execute the Lines program, and notice that now the colors are not distorted, but the program utilizes only 16 colors (see Figure 18.5).

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



**Figure 18.5.** *Using 16 colors in the Lines program.*

Note the trade off that you made in the preceding modifications. Your program can now display only limited amounts of colors. But at least you can be sure that your program will look right to all your users.

# 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](mailto: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](mailto:tegosoft@msn.com)

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

=====

**TegoSoft Visual Basic 4 Sel Study Tutorial - Lesson 18**

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