

**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 26 - Combo Boxes


In this lesson you'll learn to write programs that utilize ComboBoxes. The material that is presented in this lesson is very similar to the material that you learned in Lesson 25 (List Boxes). However, the differences and the advantages of using ComboBoxes are outlined.


## 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\Lesson26** 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 **MyCombo.FRM** inside the **C:\VBMyProg\Lesson26** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **MyCombo.VBP** inside the **C:\VBMyProg\Lesson26** 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 MyCombo Program

You'll design the window of the MyCombo program so that it will look as shown in Figure 26.1.

 Set the properties of Form1 as follows:

**Name:** Form1

**Caption:** The MyCombo Program

**BackColor:** White

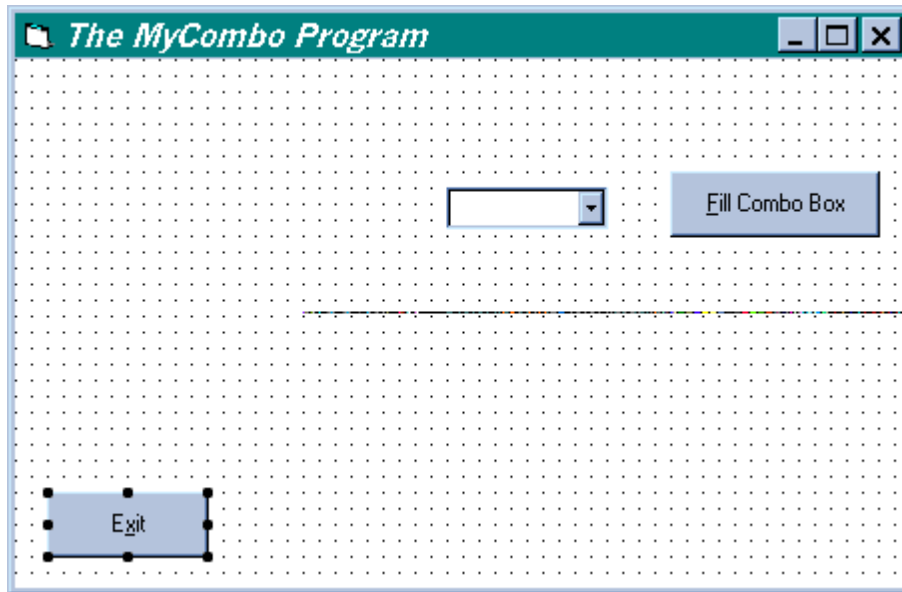



Figure 26.1. *The MyCombo program*

## 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 MyCombo program terminates itself.

## Placing a ComboBox Control

You'll now place a **ComboBox** control inside Form1.



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

Visual Basic responds by placing a ComboBox control inside Form1.



### NOTE

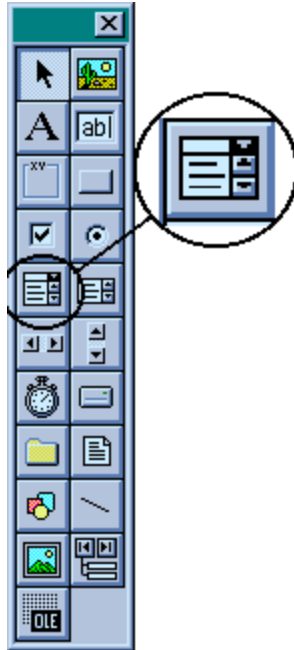
The icon of the **ComboBox** control inside the Toolbox window of Visual Basic is shown in Figure 26.2.

In Figure 26.2 the icon of the ComboBox control is shown as the fifth row from the top on the left column. However, in your toolbox, the icon of the ComboBox control may be located in a different location.




### NOTE

When you place the mouse cursor (without clicking) on the ComboBox icon inside the Toolbox, a yellow rectangle appears with the text **ComboBox** in it.



**Figure 26.2.** *The Toolbox window of Visual Basic showing the icon of the ComboBox control*

 Make sure that the **Name** property of the ComboBox control that you placed inside Form1 is **Combo1**

 The default value of the **Text** property of the ComboBox is **Combo1**. Make the **Text** property of the ComboBox empty.

### **NOTE**

The ComboBox control is very similar to the ListBox control. The difference between the ComboBox control and the ListBox control is that the Combo box is a combination of a ListBox control and a TextBox control (hence the name ComboBox). So a ComboBox has a list (just like the ListBox) that the user can select items from the list, add items to the list, and remove items from the list. The ComboBox also has a text box, and the user can type text inside this text box.

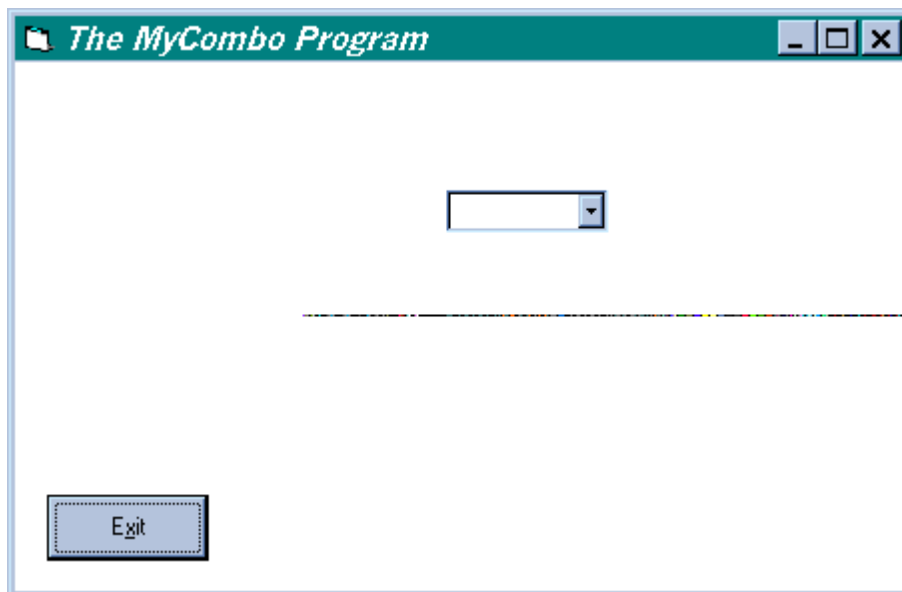
You were instructed to set the value of the **Text** property of the ComboBox to null, so upon starting the MyCombo program, there will be no text inside the text box portion of the ComboBox control.

Although you did not yet complete the MyCombo program, let's see the ComboBox control in action:



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

 Execute the MyCombo program.

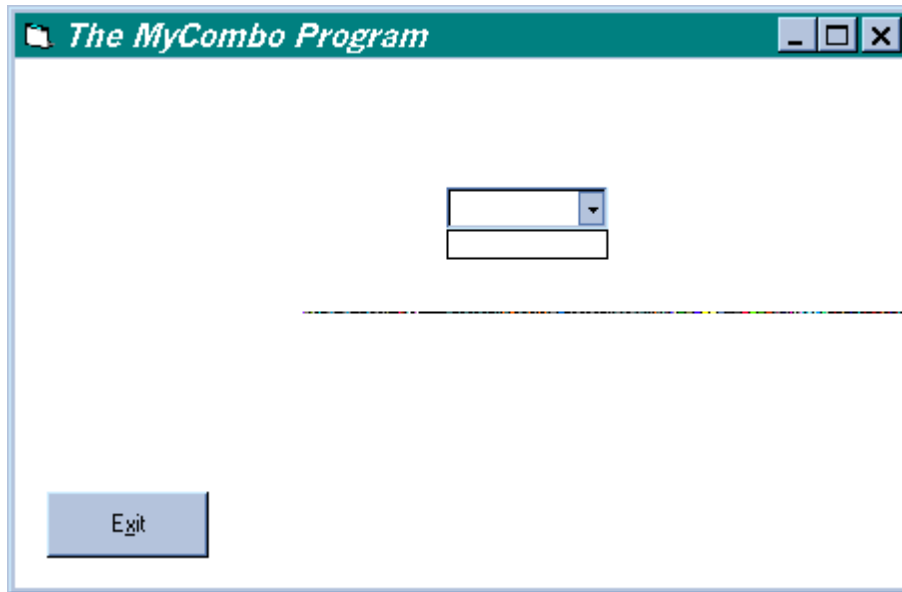
The window of the MyCombo program appears as shown in Figure 26.3.




**Figure 26.3.** *The ComboBox inside the window of the MyCombo program.*


 Click the down arrow icon (  ) of the ComboBox control.

The ComboBox control responds by dropping down an empty rectangle that serves as the container for the list of the ComboBox (see Figure 26.4). Because currently the list is empty, there are no items in the list.




**Figure 26.4.** *Dropping down the empty list of the ComboBox.*

 Type something inside the text box portion of the ComboBox to verify that this text box behaves like a regular text box .

 Click the Exit button of the MyCombo program to terminate the program.

## Placing the Fill Combo Button

You'll now place a CommandButton inside Form1. This button will serve as the mechanism to fill the list of the **Combo1** list box.


 Place a CommandButton inside Form1, and set its properties as follows:

**Name:** cmdFillComboBox

**Caption:** &Fill Combo Box

## Filling the Combo Box

You'll now write code that fills the **ComboBox**.

 Type the following code inside the **cmdFillComboBox\_Click()** procedure:

```
Private Sub cmdFillComboBox_Click()  
  
Dim ItemsOfComboBox(0 To 5)  
Dim Counter  
  
ItemsOfComboBox(0) = "Item 0"  
ItemsOfComboBox(1) = "Item 1"  
ItemsOfComboBox(2) = "Item 2"  
ItemsOfComboBox(3) = "Item 3"  
ItemsOfComboBox(4) = "Item 4"  
ItemsOfComboBox(5) = "Item 5"  
  
For Counter = 0 To 5  
    Combo1.AddItem ItemsOfComboBox(Counter)  
Next  
  
End Sub
```

You declared an array with 6 elements in it:

```
Dim ItemsOfComboBox(0 To 5)
```

You declared a local variable:

```
Dim Counter
```

You filled the elements of the array:

```
=====
```

```
ItemsOfComboBox(0) = "Item 0"  
ItemsOfComboBox(1) = "Item 1"  
ItemsOfComboBox(2) = "Item 2"  
ItemsOfComboBox(3) = "Item 3"  
ItemsOfComboBox(4) = "Item 4"  
ItemsOfComboBox(5) = "Item 5"
```

Finally, you use a **For()** loop to execute the **AddItem** method:



```
For Counter = 0 To 5  
    Combo1.AddItem ItemsOfComboBox(Counter)  
Next
```

After executing the preceding **For()** loop, the list of the ComboBox is filled with the contents of the elements of the **ItemsOfComboBox** array.



Let's see your code in action:


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


 Execute the MyCombo program.

 Click the down arrow icon (  ) of the ComboBox to expand the list, and verify that the list is currently empty.

 Click the **Fill Combo Box** button.

 Click the down arrow icon (  ) of the ComboBox to expand the list, and verify that now the list contains items.

 Click any of the items in the list of the ComboBox, and note that the item that was clicked is copied to the text box section of the ComboBox.

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



## NOTE

If the ComboBox is not high enough to fit all the items, a scrollbar is automatically attached to the ComboBox so that the user can scroll the list.

## What You Accomplished in This Lesson



You completed Lesson 26 of the Self Study Visual Basic tutorial. In this lesson you learned to write programs that utilize ComboBoxes. As you saw, ComboBoxes are very similar to List Boxes. However, a ComboBox has a text box on its top.

## Frequently Asked Questions



**Q1.** Should I use a List Box or a Combo Box in my programs?

**A1.** It depends on what you are trying to achieve. Sometimes, you'll want your user to "pick" an item from a pre-prepared list, and you will not want the user to "pick" an item that does not appear already in the list. In this case, the List Box is the appropriate mechanism to implement this feature.

On the other hand, sometimes you will want your user to pick up an item from a list, and in addition, you may want your user to be able to type his/her own item (an item that does not appear in the list). In this case, the ComboBox is the appropriate tool to use. In the Project section of this lesson you'll implement a mechanism that uses the ComboBox for picking

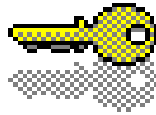
an item from the list of the ComboBox as well as letting the user type his/her own item.

## Exam



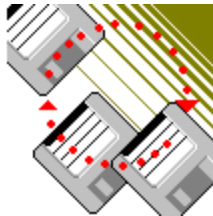
1. The ComboBox is called a ComboBox because it is a combination of: \_\_\_\_\_ and \_\_\_\_\_.

## Answers to Exam




1. The ComboBox is called a ComboBox because it is a combination of: a **List box** and a **Text box**.

## Project



Make the following modifications to the program:





 Copy the entire code that appears inside the **cmdFillComboBox\_Click()** procedure into the **Form\_Load()** procedure.


Your **Form\_Load()** procedure should now look as follows:

```
Private Sub Form_Load()  
  
Dim ItemsOfComboBox(0 To 5)  
Dim Counter  
  
ItemsOfComboBox(0) = "Item 0"  
ItemsOfComboBox(1) = "Item 1"  
ItemsOfComboBox(2) = "Item 2"  
ItemsOfComboBox(3) = "Item 3"  
ItemsOfComboBox(4) = "Item 4"  
ItemsOfComboBox(5) = "Item 5"  
  
For Counter = 0 To 5  
    Combo1.AddItem ItemsOfComboBox(Counter)  
Next  
  
End Sub
```

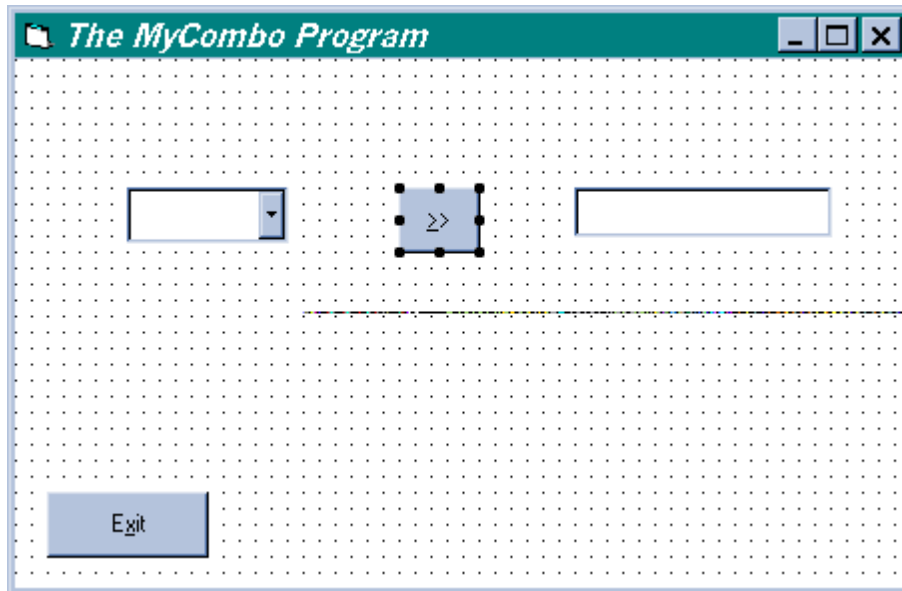
So upon starting the program, the list of the ComboBox will filled with the values of the elements of the array.

 Delete the **cmdFillComboBox** button from Form1.


 Execute the MyCombo program, and verify that upon starting the program, the list of the ComboBox is already filled with items (you have to click the down arrow icon () of the ComboBox to drop down the list of the ComboBox to see the list).

 Experiment with the MyCombo program. Then click its Exit button to terminate the program

You'll now implement a mechanism that lets the user select an item. When you'll complete placing the controls that this mechanism needs, Form1 will look as shown in Figure 26.5.



**Figure 26.5.** Form1 after placing the Label and the Copy To button.

 Move the ComboBox to the left as shown in Figure 26.5.

 Place a label control inside Form1 and set its properties as follows:


**Name:** lblChoice

**BorderStyle:** 1-Fixed Single

**Caption:** Make it empty

**Alignment:** 2-Center


**BackColor:** White


 Place a CommandButton inside Form1, and set its properties as follows:

**Name:** cmdTransfer

**Caption:** &>>


You'll now implement a mechanism that performs the following:


 The user selects an item from the list by clicking it. As you know by now, when an item in the list of the ComboBox is clicked, that item is copied to the text box section of the ComboBox. Now that the selected item appears inside the text box of the ComboBox, the user can click the **cmdTransfer** button to transfer the item to the **lblChoice** Label control.

 The user can type something inside the Text box of the ComboBox and then click the **cmdTransfer** button. As a result, the text that the user typed will be transferred to the **lblChoice** label.

 Attach the following code to the **cmdTransfer** button:

```
Private Sub cmdTransfer_Click()  
  
    lblChoice.Caption = Combo1.Text  
  
End Sub
```

 Select **Save Project** from the **File** menu of Visual Basic to save your work, then execute the MyCombo program.


 Click the down arrow icon of the ComboBox to display the list of the ComboBox, and then click on one of the items.

The ComboBox responds by displaying the clicked item inside the text box section of the ComboBox.


=====

 Click the **cmdTransfer** button.

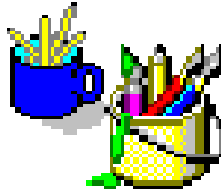
The MyCombo program responds by copying the contents of the Text box of the ComboBox to the **lblChoice** label.

 Type something inside the text box section of the ComboBox, and then click the **cmdTransfer** button.

The MyCombo program responds by copying the contents of the text box of the ComboBox to the **lblChoice** label.

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

## Cosmetic Considerations



The List box section of the ComboBox has all the characteristics of the ListBox control. This means that you can add items, remove items, copy paste and cut items in the same way that you implement these mechanisms for the List box. The only difference is that when you use the **TypeOf** command in the **If** and **Elseif** statements, you can detect whether the active control is the ComboBox by using statements as follows:

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

```
...
```

```
...
```

```
...
```

The preceding **If** statement is satisfied provides that the ComboBox is the active control.

=====

# Sorting


For cosmetic reasons, you may want to sort the items of the list of the ComboBox (just as you sorted the list in the List Box).

 At design time, set the **Sort** property of the ComboBox to **True**.

Now the list of the ComboBox is sorted alphabetically.

# Use Different Fonts


Almost all the controls that display text have the **Font** property. You use the **Font** property to set the font that the control uses to display text. Depending on the particular application that you are developing, sometimes it is appropriate to use larger fonts for the text and list of the ComboBox.


 Set the **Font** property of the Combo1 ComboBox as follows:

**Font:** MS Sans Serif

**Size:** 12

**Bold:** Yes

 Execute the MyCombo program and notice that the font used in the text box section and the list section of the ComboBox is the font that you set as the **Font** property of the ComboBox (see Figure 26.6).

 Experiment with the MyCombo program. Then click its Exit button to terminate the program.

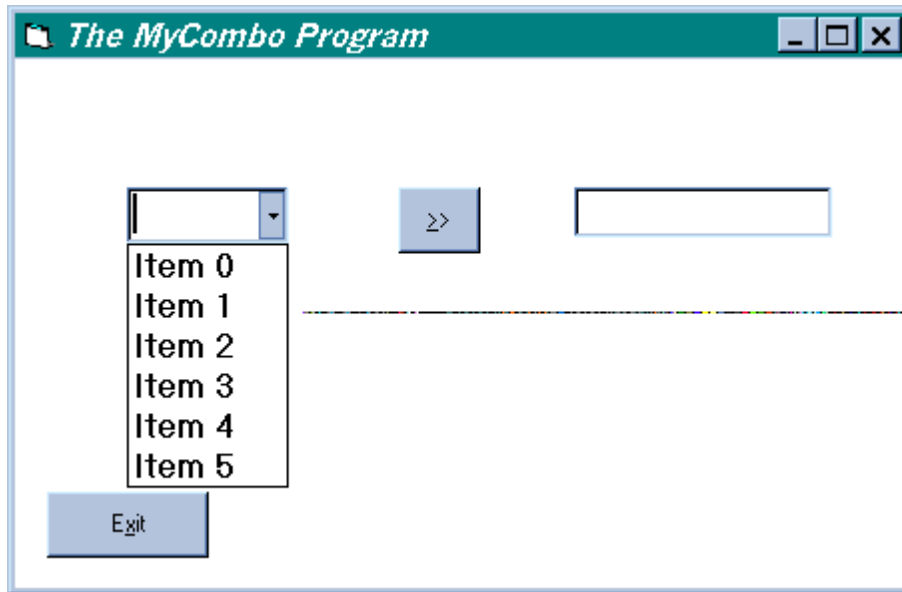


Figure 26.6. Using a larger font for the ComboBox.

## 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

**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