Chapter Six: Another Controls

6.1 Frame

A frame is a rectangular region on a form that holds other controls and groups the controls into a single set.

When you want to place multiple sets of option buttons on a form, first place the frame onto the form. (You can place any control on a frame, but the frame especially holds a group of option buttons or cheks boxes so you can offer multiple option button sets.)

The frame control does support properties that determine the frame's look and caption and a frame does support a few events, but most programmers use the frame as a holding place to group other controls. Once you place controls in a frame, you can move the frame and all the frame's controls move with it.

6.1.1 Frame Properties:

Caption	**Title information at top of frame.**
Font	**Sets font type, style, size.**
BackColor	**Specifies the frame background color.**
Enabled	**Determines whether the frame is active. e, style, and size.**
Height	**Holds the height of the frame in twips.**
Left	**Holds the number of twips from the frame left edge to the Form window's left edge.**
MousePointer	**Determines the shape of the muse cursor when the user moves the muse over the frame.**
ToolTipText	**Holds the text that appears as a tooltip at runtime.**
Top	**Holds the number of twips from the frame top edge to the Form window's top edge.**
Visible	**Determines whether the frame appears or is hidden from the user.**
Width	**Holds the width of the command button in twips.**

6.2 Option buttons

Option buttons work as a group, only one of which can have a True (or selected) value.

A frame allows you to create a group of option buttons in it. This is particular useful when you want to use option buttons in your program. You should create a group of these buttons so that the user can select one.

6.2.1 Option buttons properties

Alignment	Determines whether the label's caption appears left-justified, centered, or right-justified within the label's boundaries.
BackColor	Specifies the command button's background color.
Caption	Holds the text that appears on the command button.
Enabled	Determines whether the command button is active
Font	Produces a Font dialog box in which you can set the caption's font name, style, and size.
Value	Indicates if selected (True) or not (False). Only one option button in a group can be true.
MousePointer	Over the command button.
Picture	Holds the name of an icon graphic image that appears on the command button as long as the Style property is set to 1-Graphical.
Style	Determines whether the command button appears as a standard Windows command button (if set to 0-Standard) or a command button with a color and possible picture (if set to 1-Graphical).
ToolTipText	Holds the text that appears as a tooltip at runtime.
ForeColor	Holds the color of the text box's text.
Visible	Determines whether the label appears or is hidden from the user.

6.2.2 Option Button Events

Click	Triggered when a button is clicked. Value property is automaticallyChanged by Visual Basic.

Example (1):- Design the user interface contains **text1**, the **frame** and **two options** grouped in the frame and set the following properties below. And then bold the font of the text in **textbox**, if the user chooses the Bold or make the font of text italic if the user chooses Italic.

1. **The properties of the objects**

Text1	
Name	**txtstring**
Text	**blank**
Lable1	
Caption	enter the string
font size	14
Autosize	**true**
Frame	
Caption	main menu
Forecolor	red (&H000000FF&)
option1	
caption	bold
font size	14
option2	
caption	Italic
font size	14

So the user interface as shown below

2- And then attach the following code to the form1 and option1 and option2

Private Sub Form_ Activate ()

Option1.Value = False

Option2.Value = False

End Sub

……

Private Sub Option1_Click()

If Option1.Value = True Then

Txtstring.FontBold = True

End If

End Sub

.....

Private Sub Option2_Click()

If Option2.Value = True Then

txtstring.FontItalic = True

End If

End Sub

6.3 Check boxes

The Check Box control works just like the option button, with two differences: A selected check box shows the selection with a checkmark so the user knows that a choice is made when a check mark appears inside of the corresponding square, the user can select one or more check boxes even if those check boxes reside in the same frame or on the same form

6.3.1 Check box properties

The Check Box control properties just like the option button properties except the value properties, so see the properties of option buttons. The value properties of check box as shown below

Value: - Indicates if unchecked (0, vbUnchecked), checked (1, vbChecked) .

6.3.2 Check Box Events

Click	Triggered when a box is clicked. Value property is automatically changed by Visual Basic.

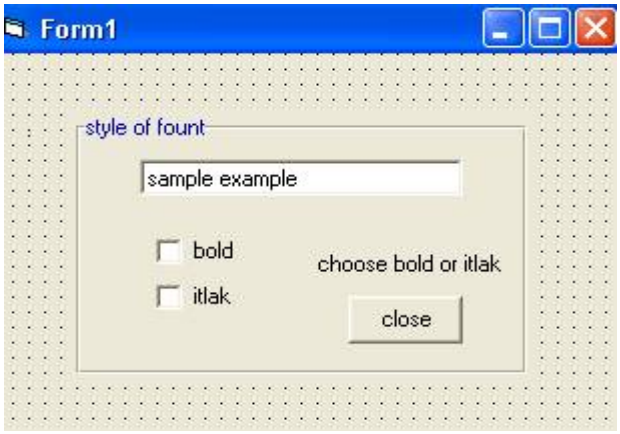
Example (2):-Design user interface contains the frame that grouped inside it the following Objects textbox to enter the string, label1, two check boxes to choose either bold font of the string or Italic font of the string in textbox, and command box to terminate the program.

1- Set the following properties to the objects that used in the form.

Frame	
Caption	style of the fount
Text1	
Text	simple example
Name	txtdisplay

Label1	
Caption	choose Bold or Italic
Autosize	true
Check1	
Name	**chkbold**
Caption	**Bold**
Check2	
chkitalic	name
Caption	Italic
Command1	
Name	cmdexit
Caption	**close**

So the user interface as shown below



2- Write the code of chkbold as follows

```
Private Sub chkbold_Click()

 If chkbold.Value = 1 Then

    txtdisplay.FontBold = True

End Sub
```

3- Write the code of chkitalic

Private Sub chkitlak_Click()

If chkitlak**.**Value = 1 Then

txtdisplay**.**FontItalic = True

End Sub

4- write the code of cmdexit

Private Sub cmdexit_Click()

End

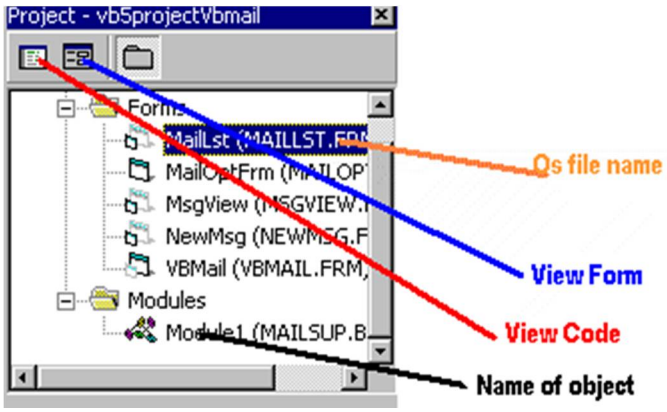
End Sub

SO when creating any visual basic application there are four steps.

1. Create a visual interface
   - create a form (VB term for a window)
   - add **controls** to form
   - edit **the properties** of each control
2. Write the code
   - write code related to the objects(event procedure)
   - writ code not related to the objects (general procedure)
3. Save and run the program.
4. Build an executable file.

Viewing the project in detail.

Once a project is loaded, the name of the form(s) that it contains is displayed in the project window. To view a form in design mode, select the form required by clicking with the mouse to highlight its name, then clicking on the view form button.



Making your project as executable file

To make an executable from a project choose 'Make project.exe ‘from the File menu. Then click once on the Make project.exe choose a default location to store your executable .After that the file execute without need to open visual basic program.

6.4 List box

A list box displays a list of items from which the user can select one or more items. If the number of items exceeds the number that can be displayed, a scroll bar is automatically added.

Each item in a list box has a subscript just as each element in an array has a subscript. The first item that you add to a list box has a subscript of 0, the second has a subscript of 1, and so on.

The user cannot add items to a list box at runtime, only the user can scroll and select items from a list box at runtime.

6.4.1 The properties of the list box

Table1: The basic list box properties.

Property	Description
BackColor	Specifies the list box's background color.
Columns	Determines the number of columns. If 0, the list box scrolls vertically in a single column. If 1 or more, the list box items appear in the number of columns specified (one or more columns) and a horizontal scrollbar appears so you can see all the items in the list.
ForeColor	Specifies the list box's text color.
Height	Indicates the height of the list box in twips.
Appearance.	Selects 3-D or flat appearance.
List	Set the items in list box at design time. You can enter only one at a time, and most programmers usually prefer to initialize the items in the list box at runtime.

MultiSelect	The state of the list box's selection rules. If 0-None (the default), the user can select only one item by clicking with the mouse or by pressing the Spacebar over an item. If 1-Simple, the user can select more than one item by clicking with the mouse or by pressing the Spacebar over items in the list. If 2-Extended, the user can select multiple items using Shift+click and Shift+arrow to extend the selection from a previously selected item to the current one. Ctrl+click either select or deselect an item from the list.
ListCount	Number of items in list. This property appears at run time
ListIndex	The number of the most recently selected item in list. If no item is selected, ListIndex = -1. This property appear at run time
Selected	Set equal to True or False, depending on whether corresponding list item is selected. This property appear at run time
Style	Determines whether the list box appears in its usual list format or, with check boxes to the selected items, in this case the multiselect must be 0.

Note:-

When placing a list box control on the form, decide how tall you want the list box to be by resizing the control to the size that fits the form best. Remember that if all the list box values don't all fit within the list box, Visual Basic adds scroll bars to the list box so that the user can scroll through the values.

6.4.2 List box Event

Table.2 contains the basic list box events that you can use in a program. You'll rarely write event procedures for list box controls, however. Most of the time, you'll let the user scroll through the list box values to see information they need; programs don't need to respond to list box events as often as they need to respond to command buttons and text boxes.

Table.2. the list box control's events.

Event	Description
Click	Occurs when the user clicks the list box control.
DblClick	Occurs when the user double-clicks the list box control.

6.4.3 List box control methods

Table 3 contains a list of list box control methods that you'll need to use for initializing, analyzing, and removing items from a list box control. Methods works like miniature programs that operate on controls. Here is the format of a method's use on a list box: **Listbox name. Method**

The control name always precedes the method and the dot operator.

Table.3. List box methods.

Method Name	Description
AddItem	Adds a single item to the list box.
Clear	Clears all items from the list.
RemoveItem	Removes a single item from a list box.

Use the **AddItem method** to add a single item to a list box control. Suppose that you want to add a few state names to a list box named **lstStates**. The following code adds the state names:

```
'Add several states to a list box control
lstStates.AddItem "New York"
lstStates.AddItem "California"
```

```
lstStates.AddItem "Nebraska"
lstStates.AddItem "Florida"
lstStates.AddItem "Nevada"
lstStates.AddItem "New Mexico"
```

This code would most likely appear in the **Form_Load() event procedure** so that the list boxes are initialized with their values before the form appears and before the list boxes are seen on the form.

Note:-

The list box acts a little like an array.

Each item in a list box has a subscript just as each element in an array has a subscript. The first item that you add to a list box has a subscript of 0, the second has a subscript of 1, and so on. To remove the third item from the list box, therefore, your code can apply the **RemoveItem method**, as follows:

```
lstStates.RemoveItem(2)          ' 3rd item has a subscript of 2
```

Keep in mind that, as you remove items from a list box, the remaining item subscripts adjust accordingly. Therefore, if a list box contains seven items, each item has a subscript that ranges from 0 to 6. If you remove the fourth item, the list box items will then range from 0 to 5; the subscript of 5 will indicate the same item that the subscript of 6 indicated before the RemoveItem method removed the fourth item.

If you want to remove all items from the list box, use the **Clear method** . The following simple method removes all the state names from the state list box:

```
lstStates.Clear                  ' Remove all items
```

You can assign individual items from a list box control that contains data by using the **List property**. You must save list box values in **string** or **variant variables** unless you convert list box items to a numeric data type using Val () first. The following assignment statements store the first and fourth list box item in two string variables:

FirstStringVar = lstStates.List(0)
SecondStringVar = lstStates.List(3)

We can use the **if** – **then** statement to test the value of certain item in the listbox as shown below

```
If lstStates.List(0) = "New York"  Then
        Text1.Text = "beautiful city"
    End If
```

This statement means if the first item is equal to " New York " then put the string "beautiful city "in textbox1

We can use the **ListIndex property** to returns the subscript of certain item in the listbox when selected this item from list box. The following assignment statement store the subscript of the selected item in the listbox in a numeric variable named loc:

Loc=lstStates . ListIndex

If the user select the frist item in the list box, then the **ListIndex** is equal to zero, because the subscript of the frist item is equal to zero. So the loc is equal also zero

We can use the following statement to check the selected item in the listbox by using the **If - statement** as shown below

If lstStates **. ListIndex** = 1 then

Text1.text = **"crowed city"**

End if

This statement means, if the user select the second item from listbox, then put the string "crowed city "in the textbox1

The **Selected** property returns either a **true** or **false** value that determines whether a user has selected a list box item. The Selected property returns **true** for possibly more than one list box item if the **MultiSelect** property is set to either **1-Simple** or **2-Extended**. Those properties indicate that the user can select more than one item at once. See the following code statement

If lstStates.Selected**(0)**= **true** And **lstStates . Selected (1)** = **True** Then

Text1.text =" crowed and beautiful cities "

End If

This statement means if the user selected the first item and the second item from list box at the same time, then put the string "crowed and beautiful cities "in the textbox1.

Note:-

The **listIndex** and **selected** properties used to check for selected items in the **list box** control.

The **ListCount** property always provides the total number of items in the list box control. For example, the following statement stores the number of list box items in a numeric variable named Num:

Num = lstStates.ListCount

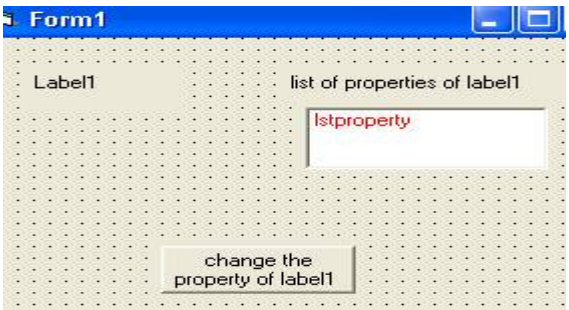
Example(3):-Write the application in vb. that contains the listbox named lstproperty that hold the names of three label properties(Appearance , backcolor , caption), these item write in the listbox in the form-load event procedure . And add the command 1 named cmdchange and its caption is 'change the property of label1" to check for selected items in list box, that means if the user select the first item ,change the appearance property of lable1 and so on .

1- Design the user interface, we need to the following controls (listbox, command1, label1 that change its properties, label2)

2- Set the following properties for the controls that used in the form.

label1	
don't change any property	
label2	
caption	list of properties of label1 'put this label above the listbox
listbox1	
name	lstproperty
forecolor	red
multiselect	0-none
command1	
name	cmdchange
caption	change the property of label1

So the user interface after design as shown below



3- Writing the code

First You'll often initialize a list box items in the Form_Load() event . The following code writes three names of label properties as shown below:

```
Private Sub Form_Load( )
lstproperty.AddItem "Appearance”
lstproperty.AddItem "BackColor”
lstproperty.AddItem "Caption”
End Sub
```

Then write the code of cmdchange-event procedure to check for selected item in the list box as shown below:

```
Private Sub cmdchange_Click( )
If    lstproperty.Selected(0) = True Then
 Label1.Appearance = 0
 ElseIf   lstproperty.Selected(1) = True Then
 Label1.BackColor = vbGreen
 Else
 "Label1.Caption = "welcom in visual basic
 End If
End Sub
```

The user interface after running the program as shown below

6.5 Combo Boxes

The combo box is similar to the list box. The differences are a combo box includes a text box on top of a list box and only allows selection of one item, whereas the user in listbox can select one or more than one of item depending on the **MultiSelect** property.

Visual Basic supports three kinds of combo boxes determined by the **Style** property.

Here are the three kinds of combo boxes:

1. **Drop-down combo box:-**Takes up only a single line on the form unless the user opens the combo box (by pressing the combo box's down arrow) to see additional values. The user can enter additional items at the top of the drop-down combo box and select items from the combo box.
2. **Simple combo box: -** Displays items as if they were in a list box. The user can add items to the combo box list (whereas the user cannot with a normal list box).
3. **Drop-down list box:-**Does not let the user enter new items, so is similar to a list box. Unlike a list box, however, the drop-down list box normally appears closed to a single line until the user clicks the down arrow button to open the list box to

its full size. Technically, drop-down list boxes are not combo box controls but work more like list boxes. The reason drop-down list boxes fall in the combo box control family is that you place drop-down list boxes on forms by clicking the combo box control and setting the Style combo box property.

6.5.1 Combo Box Properties

The combo box properties are nearly identical to those of the list box, (see the properties of list box), with the deletion of the MultiSelect property and the addition of a Style property as shown below.

Style	Determines the type of combo box your application needs. If 0-DropDown Combo, the combo box is a drop- down combo box. If 1-Simple Combo, height the combo box turns into a simple combo box that remains open to the you use at design time. If 2-DropDown List, the combo box turns into a drop-down list box that remains closed until the user is ready to see more of the list.

Note:-

That there is **no MultiSelect combo box property** as there is with **list box controls**. So the user can select only one combo box item at any one time.

Note:-

There is no **selected property** that used to check for selected items in the Combo box control, but there is only **ListIndex property** that used to check for selected item in the combo box

6.5.3 Combo Box Events:

Also the **combo box** support the **same event** that the **list box control** support. (See **the** table below).

Click	Event triggered when item in list is clicked.
DblClick	Event triggered when item in list is double-clicked. Primary way used to process selection.

6.5.4 Combo Box Methods

The **combo box controls** support the same **methods** that the **list box controls** support. Therefore, you can **add, remove**, and **clear** items from the combo box. (See the table below).

AddItem	Allows you to insert item in list.
Clear	Removes all items from list box.
RemoveItem	Removes item from list box, as identified by index of item to remove

Example(5):-Write the code to the **command control event procedure** named cmdadd to add countries names to the combo box control named comStates.

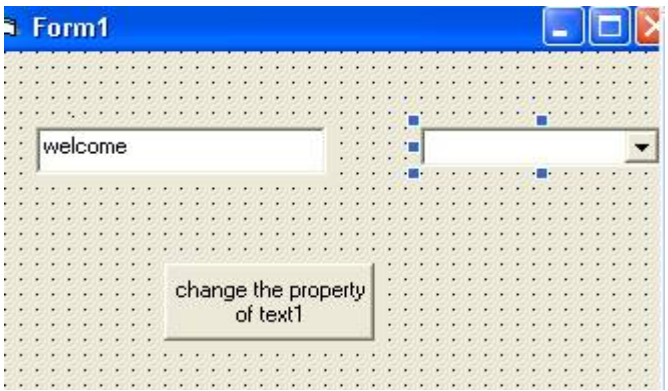
```
Private sub cmdadd _ click
comStates.AddItem "New York"
comStates.AddItem "California"
comStates.AddItem "Nebraska"
End sub
```

Example(6):-Write the application in vb. that contains the combo box named comproperty that hold the names of three text1 properties (Alignment, back color, Font), these item write in the combo box in the form-load event procedure. And add the command 1 to check for selected items in combo box that means if the user select the first item, change the Alignment property of text1 and so on.

1. Design the user interface , we need to the following controls (combo box , command1 , text1 that change its properties)
2. Set the following properties for the controls that used in the form.

text1	
text	welcome
combo box	
name	comproperty
style	0-DropDown combo
command1	
name	cmdchange
caption	change the property of text1

So the user interface after design as shown below



3- Write the code for the following objects below

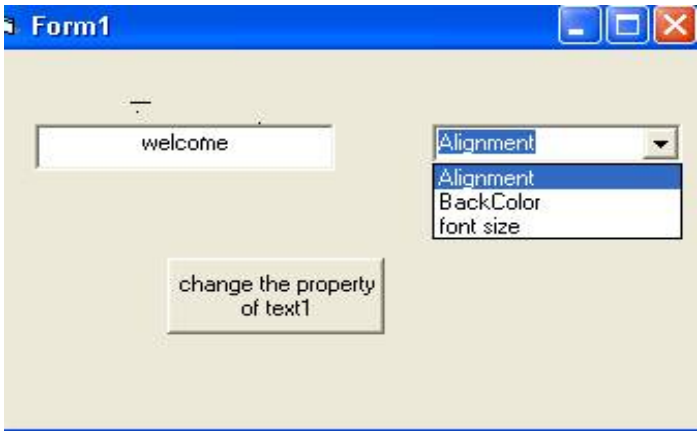
First You'll often initialize a list box items in the **Form_Load()** event . The following code writes three names of label properties as shown below:

```
Private Sub Form_Load()

Comproperty.AddItem "Alignment"

Comproperty.AddItem "BackColor "

Comproperty.AddItem "font size"

End Sub
```

Then write the code of cmdchange-event procedure to check for selected item in the combo1 box as shown below:

```
Private Sub cmdchange_Click()

If  Comproperty. ListIndex = 0 Then

 Text1.Alignment = 2
 ElseIf  Comproperty. ListIndex = 1 Then

 Text1.BackColor = vbRed

 Else

Text1.FontSize = 14

 End If

End Sub
```

So the form after running as shown below



Note:-

A list box presents users with a list of items. The user can select from the list. The user cannot add new items to the list box. If you want to present a list of items to the user and let the user enter new items, use a combo box. A combo box works a lot like a **combination list box and text box**. As users type new values into the text area and then click the appropriate command button to indicate that the text is ready, the new values go to the combo box's list.

Now we write the program to learn how can the user type new values to the text area (top of the combo box) and then click the command button to indicate that the text is ready and go to the combo box's list. The program contains combo box and command1. The items in the combo box are the following colors(red , yellow) , You'll often initialize a combo box's items in the **Form_Load()** event , and then the user type new value (green) color to text area . The code of command1 is accept this value and go to the combo box's list.

Set the following properties for the controls that used in the form

combo1	
name	comcolor
command1	
caption	Accept new value and go it to the combo box

Private sub form – load click()
Comcolor **.** AddItem "red "

Comcolor . **AddItem "yellow"**
End sub

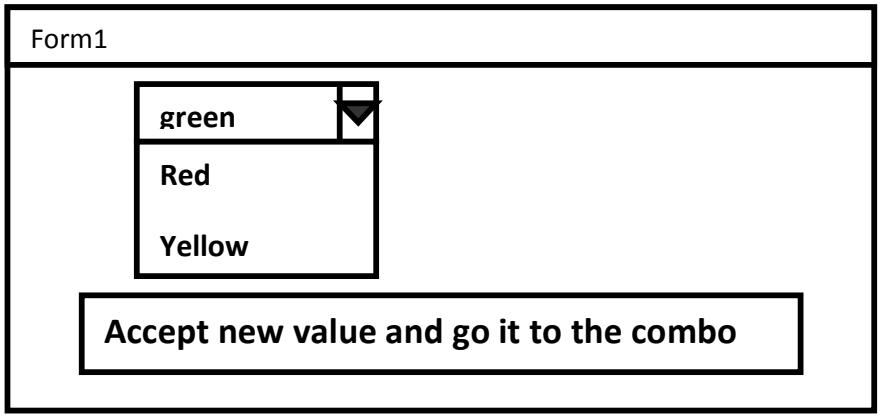
……

Private sub command1- click()

Dim co as string
co = Comcolor.Text ← The new value that entering in the text area (top of the combo box) at the run time.

Comcolor**.** AddItem co

End sub



← The form after running