Unit 4 Windows Applications in VB.NET

Structure:

4.1 Introduction

In the previous unit we discussed the conditional and looping statements that are require developing an application in VB .NET. We also explored the data types and the operators that are supported by VB .NET

In this unit we are going to discuss the windows application in VB.NET, Windows applications are supported with the windows controls, all these controls are basically a graphical object. These graphics objects help the user to interact with an application in a friendly way. You have variety of controls that satisfy the situational demand. Selection and handling of these controls are left to the programmer's choice. Toolbox has most of these controls to support the developer to design the application. In this section we are going to discuss various tool box controls with its syntax and construct.

Objectives:

After studying this unit, you will be able to:

- develop application using textbox, Button and Label controls
- declare and define the checkbox and Radio Buttons
- explain the role of List and Combo controls
- construct the Image List, Tree and List views
- discuss and design the Tool, Status and Progress bars for an application

4.2 Working with TextBox, button and labels

Whenever you want to create an application in VB .NET will allow you to create a new project as we discussed in the previous units. Thus default project creation comes with the default blank form1. Form is acting as a container, where we are going to drag and drop controls to design an application. As shown in figure 4.1 the default form1 will be loaded. You can observe from the figure 4.1 the form is loaded with the default values like title bar at the top displays the name of the form. Title bar are also listing the control box with max, min and close buttons. Form's properties can be changed either through the property window or through the coding. To change the name of the form title, go to property window set Text property to My Form or any name of user's choice. In case if you want to change the back color of the form to yellow through coding below is the syntax.

```
Form1.BackColor=Color.Yellow
```

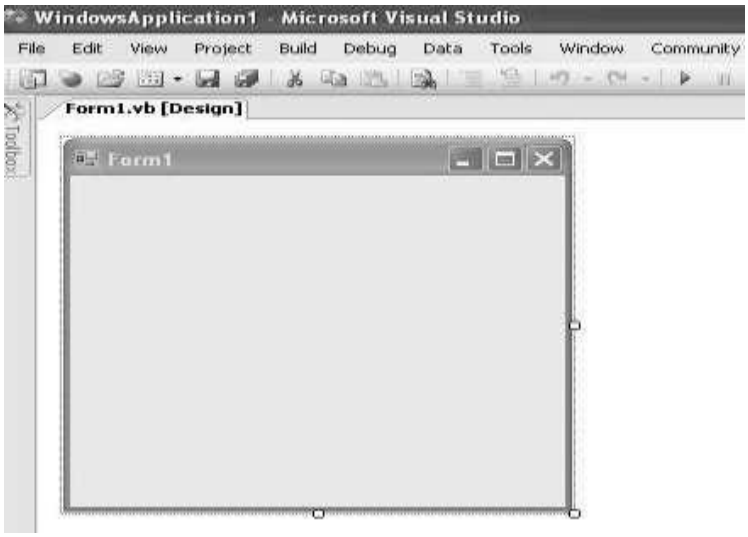

Fig. 4.1: Windows Form

TextBox

TextBox controls can be added to the forms by dragging and dropping into the form. The properties of the TextBox can be changed through the property window or through coding. If you double click the control you can reach the event handler namely Form1_Load etc. Also the code will have the list of event pertaining to the selected control. TextBox control is one of the most used controls used to display text or to add text on the form. Since
we are discussing the first control on the VB .NET application we will see the list of properties, methods and the events that are available for TextBox control.

Below listed are the few properties with its descriptions. **TextAlign:** used for aligning text in the TextBox control. **Multuipline:** Set to accept more than one line.

ScrollBars: To set vertical and horizontal scroll bars.

MaxLength: Restricts the maximum number of characters that the TextBox can accept.

Enabled: set the enability True or False. **Index:** Indicates the index in the control array.

Read Only: Restricts the user to access the control by setting True/False.

Following are the list of TextBox methods with its descriptions. **Drag:** Supports Drag and Drop functions

Focus: Keep the cursor in that and set the focus to that control.

Set Bounds: Indicates the boundary of the control about location and size. **Clear:** Clear the content or text from the TextBox control

Select: Helps to select the text from the TextBox **SelectAll**: Selects all the text from the TextBox **Cut:** Selected Text can be cut from the control

Copy: Selected Text can be Copied

Paste: Helps to paste the content to the clip board.

Following are the list of events pertaining to the control especially the TextBox control.

AutoSizeChanged : Where ever there is change in the AutoSize property this property will be triggered.

ReadOnlyChanged: This event will be triggerd when ReadOnly property gets updated

Click: When the object get clicked this event will be triggered.

KeyDown: Used for validating text triggered any key pressed on the text Box.

Buttons

When you click on the windows form button it performs some action. Whenever you click on the button you can feel the button is pressed in and released. When the user clicks the button the button event is triggered and performs the action based on the choice. The text you want to display on the button control can be updated with the text property. Font and Text Algin are the supportive property to design the button control. You must have observed buttons are designed by the programmer for various operation using different handler. Now we will see one special use in button control is, this button can be used to cancel the form this can be achieved by setting the Cancel Button property to the button control name. Whenever the user is pressing the Esc key the cancel key will be triggered helps the user to come out from the application easily.

Labels

Label control is one of the most used controls almost in all the forms. Label control can be used to accept the descriptive text and where the users need not to use the content. Text property can be changed to change the text content of the label box. Basically label controls are used in the forms as a indicators to the user to guide them to give the appropriate input. Using the label properties control you can change the location enability, size, color etc. Instead of plian text you can load an image to the lable control using the image property.

4.3 Working with Check Box and Radio Buttons

Check box and radio buttons are the two controls used to list the multiple choices for the user.

Check Box

The check box is one of the important controls to make the statement True or False. You can see the check box in the figure 4.2 shows a small box, this helps the user to say true or false by selecting or deselecting the specific choice. This small box in the check box represents the two statuses when the check box is ticked with right symbol means the status is true, if it is empty the status is false. In the given figure the check box is selected so the status is true.

Fig. 4.2: Check Box Selected

Whenever you add a check box control to the form the default value of check status is false. There are three different ways to change the status of the check box.

- At run time you can change the status by selecting the check box
- You can go to property window change the "Checked" status to "True"
- Through coding we can change the status *CheckBox1.Checked ="True"*

In order to find out whether an item is selected, get the value of its Checked property. Another possibility consists of toggling the state of the check mark with regards to another action. For example, you can check or uncheck the check mark when the user clicks another button. To do this, you can simply negate the truthfulness of the control as follows:

CheckBox1.Checked = Not CheckBox1.Checked

By default, a check box control appears as a square box that gets filled with a check mark when the user clicks it. Optionally, you can make a check box control appear as a toggle button. In that case, the button would appear as a regular button. When the user clicks it, it appears down. If the user clicks it again, it becomes up.

To change the appearance of a check box, assign the Button or Normal value to its Appearance property. The Appearance values are defined in the Appearance enumerator. You can also do this programmatically as follows:

```
Private Sub Form1_Load(ByVal sender As System. Object, ByVal e As System. EventArgs) Handles MyBase.Load
    Me.CheckBox1.Appearance = Appearance. Button
End Sub
```

Radio Buttons

This button is otherwise called as option button. It should come with a group of similar such type. Individual radio button in that group can be identified with the small circle. When the particular option of the radio button gets selected means the small circle will be filled with the dark black circle. In the below figure 4.3 the option large got selected the other two or not. The option button is accompanied with the label with the appropriate title.

Fig. 4.3: Radio Buttons

If you compare both the check box and the radio buttons, both has the list of option to select. The main difference is in check box you can select more than one option that means multiple

selections are possible. Whereas radio or option buttons, the selections are mutually exclusive. Only one among the group can be selected. There are two main ways you can use this property. To select a particular radio button, set its Checked property to true. To find out if an item is selected, get the value of its Checked property. You can also programmatically check a radio button. Here is an example:

```
Private Sub Form1_Load (ByVal sender As System. Object, ByVal e As System. EventArgs) Handles MyBase.Load

    Me.RadioButton2.Checked = True End Sub
```

By default, the round box of a radio button control is positioned to the left side of its accompanying label. In Microsoft .Net applications, you have many options. Besides the left position, the most common alignment consists of positioning the round box to the right side of its label. The position of the round box with regards to its label is controlled by the CheckAlign property. The possible values are: TopLeft, TopCenter, TopRight, MiddleRight, BottomRight, BottomCenter, and BottomLeft. You can also do this programmatically as follows:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    Me.RadioButton1.CheckAlign = ContentAlignment.MiddleRight End Sub
```

Radio buttons appear by default as rounded boxes that get filled with a big dot when the user selects one. Optionally, you can make a radio button appear as a toggle button. In that case, the buttons would appear as regular buttons. When the user clicks one, it appears down while the others are up. If the user clicks another button, the previous one becomes up while the new one would be down. To change the appearance of a radio button, assign the Button or Normal value to its Appearance property. The Appearance values are defined in the Appearance namespace. The syntax for change of radio button appearance is

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    Me.RadioButton1.Appearance = Appearance.Button End Sub
```

4.4 ListBox, Combo, Picture, Scrollbars and Splitters

ListBox

ListBox control helps to display a list of items. It can have text images and other controls as its data. User can select either one or group of items from listBox controls. You can create a list box either through tool box control or at run time also. As soon as you drag and drop ListBox control in the form it appears as shown in figure 4.4.

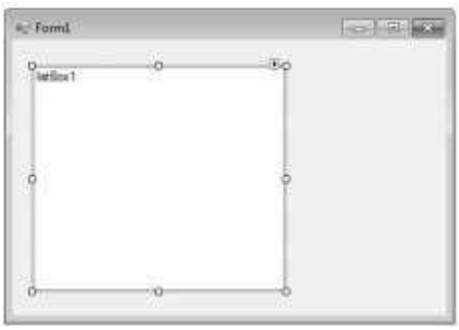

Fig. 4.4: ListBox control

Following are the few properties can be updated at the run time through coding in ListBox controls.

ListBox1.Name=”List1” – Changes the name of the control.

The following set of properties changes the width, height and location of the ListBox control.

ListBox1.Location = New System.Drawing.Point(10, 10) ListBox1.Size = New System.Drawing.Size(200, 150) ListBox1.Width = 350

ListBox1.Height = 200

ListBox1.Font = newFont("Georgia", 16) – Changes the font ListBox1.BackColor = System.Drawing.Color.Orange ListBox1.ForeColor = System.Drawing.Color.Black

List of item that you want to add in the ListBox
ListBox1.Items.Add("Mahesh Chand")
ListBox1.Items.Add("Mike Gold")

ListBox1.Items.Add("Praveen Kumar")

ListBox1.Items.Add("Raj Beniwal")

Otherwise to set the properties at the design time either press F4 or go and select properties menu, Property window appears as shown in figure 4.5. All the properties can be updated through the property window. You can see in figure 5.4, the property highlighted is **Items** and you can see collection button if you click on the button and update the list that you want to list.

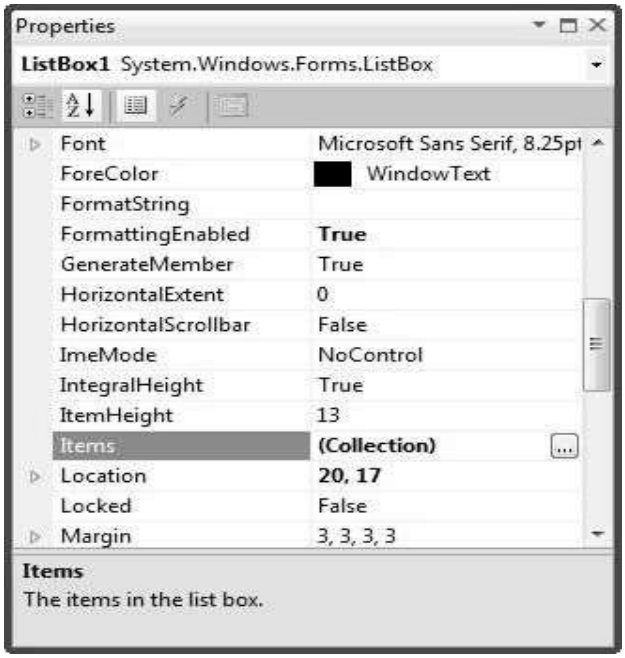

Fig. 4.5: Property window

The following syntax helps to display the selected item from the list box in the message box window.

MessageBox.Show(ListBox1.Text)

Combo Box Control

Combo Box control is used to list the group of items. It is a drag down menu, where the item will be dragged / listed down once the user click

s on the arrow button. This control pools the features of a text box and a list box. Combo Box appears as shown in figure 4.6

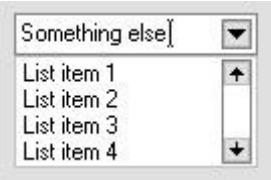

Fig. 4.6: Combo Box

This control permits the user to select an item either by typing text into the combo box, or by selecting it from the list. Similar to the ListBox in the property window, the Items can be selected to give the list of data that you want to display. Following are the set of instructions to display the selected item from the Combo Box

Dim MyVariable as String MyVariable = Combobox1.Text MsgBox MyVariable

Combo Box has the DropDownStyle property. Where you can select the combo Box styles according to the programmer wish to display the content.

Picture Box control

The picture Box control supports the following set of picture formats like bitmap, icon, GIF, JPEG etc. Image property can be set either design time or at run time to the Image that you want to display. You can programmatically change the image displayed in a picture box, which is particularly useful when you use a single form to display different pieces of information.

PictureBox1.Image = Image.FromFile("C:\testImage.jpg")

The SizeMode property, which is set to values in the PictureBoxSizeMode enumeration, controls the clipping and positioning of the image in the display area.

PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage

There are five different PictureBoxSizeMode is available to PictureBox

control.

AutoSize – Sizes the picture box to the image.
CenterImage – Centers the image in the picture box.
Normal – Places the upper-left corner of the image at upper left in the picture box
StretchImage – Allows you to stretch the image in code

You can change the size of the display area at run time with the ClientSize property.

pictureBox1.ClientSize = New Size(xSize, ySize)

The following VB.Net program shows how to load a picture from a file and display it in stretch mode.

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

PictureBox1.Image = Image.FromFile("d:\testImage.jpg") PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage

End Sub

.

Scroll bars

The scroll bar control in VB .NET can be used where you have large information are available in the form. This scroll bars used to navigate either horizontally or vertically across the form to view the information. Basically scroll bar supports two types of scroll bars called HScrollBar for horizontal scrolling and VScrollBar for vertical navigation. You can see the horizontal scroll bar listed in the left side of the figure 4.7. Right side shows the form with the horizontal and vertical bar embedded

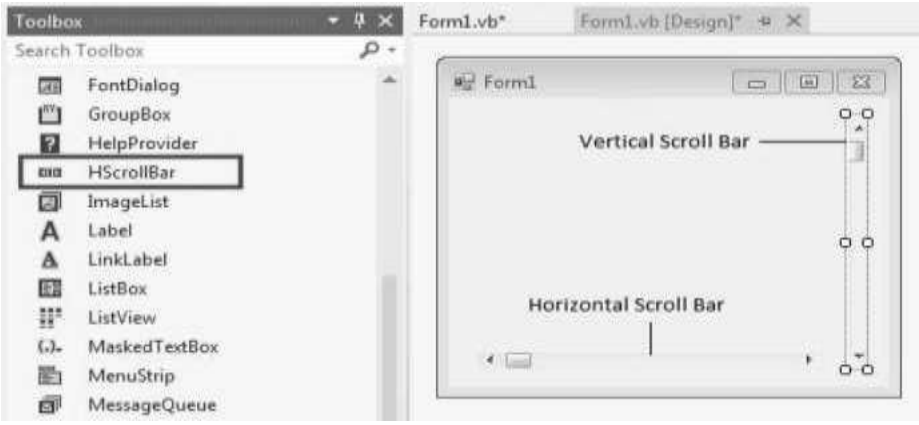

Fig. 4.7: Scroll Bars

Following are the list of properties of ScrollBar controls **AutoSize** – ScrollBar is automatically resized to fit its contents. **Backcolor** – Sets the background color of the control **Forecolor** – Sets the forecolor of the control

Maximum – Set the upper limit of the scrollable range. **Minimum** – Set the lower limit of the scrollable range.

Value – sets/gets the current position of the scroll box on the scroll bar control.

```
Private Sub Form1_Load(sender As Object, e As EventArgs) _ Handles MyBase.Load
creates a vertical and horizontal scroll bar on the form Dim hs As HScrollBar
    Dim vs As VScrollBar hs = New HScrollBar()
    vs = New VScrollBar()
    hs.Location = New Point(10, 200) hs.Size = New
    Size(175, 15) hs.Value = 50
    vs.Location = New Point(200, 30) vs.Size = New Size(15,
   175) hs.Value = 50 Me.Controls.Add(hs)
    Me.Controls.Add(vs) ' adding both bars on form
  End Sub End Class
```

Splitters

Splitter control is used basically to divide the window in to two resizable shares. It gives the option to the user to resize the pane either increase or decrease any of the portions during runtime as shown in figure 4.8. Window form frame work supports this type resizing functionality.

.

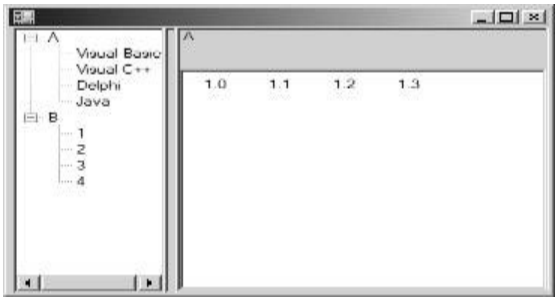

Fig. 4.8: Splitter control

This control does not move anything but the docking mechanism does this work. Usually the splitter is used between the two controls. The first control and the splitter are docked to the same edge of the window usually the left or the top. The splitter should be docked towards the inside of the window. The remaining control is then set to fill thus it does renaming space. When the user drags a splitter the splitter control only resizes the outermost control. This causes the window to perform a layout operation, recalculating the position of all docked controls.

Image List, Tree Views and List Views

In this section we are going to discuss the Image List, Tree Views and List Views controls. In visual basic, there are few controls which appear in the form and few controls that do not visible on the screen.

Image List

One such control is the image list control, supports to store the list of images. Usually you add images to an image list control at design time, using the insert picture button in the control's property pages. You can also add images list at runtime, using the Add method of its internal image collection, ListImages. To use the images in the image list you usually associate the image list with a windows common control that has an ImageList property. For each item in the common control, such as a tab in a tab strip control, you can then specify ether an index into the image lists' ListImages collection or an image's key value to associate that image with the item.

You can also reach the images in an image list with the ListImages collection's Picture property. For example if you wanted to use an image list with a control that's not a windows common control, such as a picture box, you can assign the first image in the image control to that picture box this way:

Picture1.picture=Imagelist1.ListImage(1).picture

Generally image list control tool appears in the visual basic toolbox at bottom on right side.

Tree Views

If you are user of windows explorer you must be familiar with the tree views. Tree views present data in a hierarchical way, such as the view of directions that appears in the tree at left in the windows explorer. Trees are composed of cascading branches of nodes, and each node usually consists of an image (set with the Image property) and a label (Set with the Text property). Images for the nodes are supplied by an image list control associated with the tree view control.

A node can be expanded or collapsed, depending on whether or not the node has child nodes. At the topmost level are root nodes, and each root node can have any number of child nodes. Each node in a tree is actually a programmable Node object, which belongs to the Nodes collection. As with other collections, each member of the collection has a unique Index and Key property that allows you to access the properties of the node. The Tree view control tool will be available, towards down after all the basic controls. We will see one example by adding a node Node1, to a tree view, TreeView (the tree view's Style property is set to tvwTreelinesPlusMinusPictureText, the default.

```
Private Dub Form Load() Dim Node1 As Node
Set Node1=TreeView1.Nodes.Add
Node1.Text="Node 1"
End Sub
```

If you run this code as soon as the form gets loaded the tree will be create with the Node1. You can keep adding nodes by duplicating the code with the different names like Node 2, Node 3 etc. Instead of text we can also add image to the Tree views.

List View

List view generally used to display the list of files. Each item in a list view control is itself a ListItem object and can have both text and an image associated with it. The ListItem objects are stored in the list view's ListItem collection.
List views can display data in four different view methods.

Icon mode: can be manipulated with the mouse, allowing the user to drag and drop and rearrange objects.

Small Icon mode: Allows more ListItem objects to be viewed. Like the Icon view mode, objects can be rearranged by the user.

List mode: Presents a sorted view of the ListItem objects.

Report mode: Presents a sorted view, with sub-items allowing extra information to be displayed.

You can add items to a list views ListItems collection, using its Add method each items can be added with the control. We will see one example to add three items to a list view ListView1.

```
Private Sub Form_Load()
Dim ListItem1 As ListItem
Set ListItem1 =ListView1.ListItems.Add()
ListItem1.Text="Item 1"

Dim ListItem2 As ListItem
Set ListItem2 =ListView1.ListItems.Add()
ListItem1.Text="Item 2"

Dim ListItem3 As ListItem
Set ListItem3 =ListView1.ListItems.Add()
ListItem1.Text="Item 3"
```

We need to set the ListView1 control's view property to lvwList(=2) and run this program.

Adding small icons to list view items

We will add a new image list control, ImageList2, to a program now to hold small icons. In this example we'll just place one image in ImageList2-leaf.bmp from the visual basic

common\graphics\bitmap\outline directory.

To connect the image list with the list view, right click the list view at design time, and select the properties item in the menu that appears. Click the Image List property in the property pages, and select ImageList2 in the box labeled small, then click on ok to close the property pages. Below is the code to add the images that are stored as the small icon of all the listItem.

```
Private Sub Form_Load()
Dim ListItem1 As ListItem

Set ListItem1 – ListView1.ListView1.ListItems.Add()
ListItem1.Text = "Item 1"
ListItem1.Icon=1
ListItem1.SmallIcon=1
```

Likewise you can add the items as much you want.

Finally, set the list view's view property to lvwSmllIcon=(=1) and run the program. You can see the icons we've have selected for each item displayed in the list view.

4.6 Tool Bars, Status Bar and Progress Bar

Tool Bars

Tool bar exist at the top of the window filled with buttons. Generally tool bar contains buttons correspond to items in an application menu. It provides an easy interface for the user to reach frequently used functions and commands. The user can also customize toolbars double-click a toolbar at runtime opens the customize Tool bar dialogue box, which allows the user to hide, rearrange and play buttons. To add buttons a tool bar, you add button objects to its Buttons collection, usually by working with the toolbar's property pages. Each Button can have text and image. Set text with the caption property and an image with the Image property for each Button object. At runtime you can add or remove buttons from the Buttons collection using Add and remove methods **Adding buttons to the tool bar**

You can add buttons to a toolbar control at design time by right-clicking the control and clicking the properties item in the menu that appears. When the toolbar's property pages open, click the Buttons tab. You can insert new button by clicking the Insert button. When you add a new button to a toolbar, you can associate a picture or caption with it. Each button gets a new Index value, which will be passed to the click event handler. You can also give each button a key value, which is a string that you can use and identify the button. Below you can find a syntax to identify the button index value that you selected on the toolbar.

```
Private Sub Toolbar1_ButtonClick(Byval Button As ComctLib.Button)
MsgBox "you clicked the button" & Button.Index
End Sub
```

Status Bar

Status bar generally appear at the bottom of the windows and usually hold several panels in which you can display text. Status bar gives a feedback to the user on program operation, as well other items like date of day or key states. Status bar builds around a panel's collection, which holds the panels in the status bar. You can change any the text, images or widths of any panel object using Text, Picture Width properties. To add panel objects design time, right click the status bar and click properties to display the property pages dialog box.

Adding panels to the status bar

Follow the steps to add the status panels at the design time.

- Right Click the status bar, and select the properties item in the menu that opens.
- Click the panels tab in the property pages.
- Click the insert panel button as many times as you want panels in your status bar.
- Close the property page and click ok.

Below is the code to add the panels at the run time by clicking the button

```
Private Sub Button1_Click() Dim Panel1 As Panel
Set Panel1 = Status Bar1.Panels.Add()
Panels1.Text=”Status:OK” ‘ To add the text to status bar.w
End Sub
```

We can display the images in the status bar in design time using the Picture property, pictures can be browsed through this property button thus the image gets loaded. Following code helps to load the image at the run time.

```
Private Sub Button1_Click()
StatusBar1.Panels(1).Picture=Picture1.Picture End Sub
```

Progress Bar

Progress bar gives the feedback to the user a visual feedback on What is happening during time-consuming operation. They present the user with the color bar that grows in the control to show how operation is proceeding, usually from 0 to 100 percent. The progress bar value property (not visible

at run time) determines how much of the control has been filled. The Min and Max properties set the limits of the control.

Now we are going to discuss, how we will use the progress bar actually to display data. We are going to use the progress bar's Value property to specify how much of the progress bar is visible by setting the value to Min and Max properties. Let us see an example, when the user click on the button to display progress bar whose bar lengthens from Min to Max in 10 seconds. Now you can add progress bar, button and timer control on the form. Set the timer's interval property to 1000. Set progress bar's Min value at 0 and Max property at 100 the defaults. When the form loads we disable the tuner and set the progress bar's value to 0.

```
Private Sub Form_Load()
Timer1.Enabled=False ProgressBar1.Value=0
End Sub
```

When the use clicks on the button we wanted to start the progress bar to enable the timer. We also set the progress bar back to 0.

```
Private Sub Button1_Click()
ProgressBar1.Value=0 Timer1.Enabled=True
End Sub
```

Finally in the Timer event handler, Timer_Timer() here the value of the progress bar is incremented by 10 in every 10 seconds. When the progress bar value reaches the value "100", the timer will be disabled, by that time the progression bar will complete its one single process.

```
Private Sub Timer1_Timer()
ProgressBar1.Value=ProgressBar1.Value+10
If ProgressBar1.Value >=100 Then Timer1.Enbled =False End Sub
```

When the user clicks the button the progress bar starts the motion and goes from 0 to 100 in 10 seconds

4.7 Summary

- In Windows application, Form is going to act as a container to hold various controls.
- Check box and Radio buttons are provides the user with group of choices. But check Box supports multiple choices whereas the Radio button is mutually exclusive.
- List and Combo box list the group of items that the user can select. Difference between these two is the combo has in build place holder for user's selection.
- Image list, Tree views and list views are the control does not appear on the screen.
- Tool, status and progress are the various bars that supports the developer to design and add to the form according to the need of the application.

4.8 Questions and Exercises

1. Explain the role of TextBox, buttons and labels controls with example.
2. Discuss the checkbox and the radio buttons with property list.
3. Discuss and differentiate List and Combo controls.
4. Explain how to add a button to the tool bar with an example.
5. Brief the role of Tree views.

4.9 Suggested Readings:

- http://www.vbtutor.net/VB2008Book/vb2008me_preview.pdf
- http://www.functionx.com/vbnet/Lesson03.htm
- http://vb.net-informations.com/gui/windows_forms.htm