Introduction to Visual Basic .NET

Chapter 2
Introduction to Controls, Events

VB.NET Controls

• Invoking VB.NET
• A Text Box Walkthrough
• A Button Walkthrough
• A Label Walkthrough
• A List Box Walkthrough
• The Name Property
• A Help Walkthrough
• Fonts / Auto Hide
A Text Box Walkthrough

- Drag Text Box from ToolBox
- Sizing
- Delete
- Properties
  - Text, Color, Font, Size, Location, Visible, Enabled

A Button Walkthrough

- Add the button
- Change the Text property
Add an "access key"

A Label Walkthrough

- Add the Label
- Change the Text property
- Resize the control
A List Box Walkthrough

• Add the List Box
• Add data
• Resize the control

The Name Property

• How the programmer refers to a control in code
• Name must begin with a letter
• Must be less than 215 characters long
• May include numbers and the underscore
• Naming convention: use appropriate 3 character naming prefix
  – First three letters identifies the type of control
  – Remaining letters identifies the purpose
  – E.g. a text box to store a social security number would be called
      txtSocialSecurity
Common Control Name Prefixes

<table>
<thead>
<tr>
<th>Control</th>
<th>Prefix</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>button</td>
<td>btn</td>
<td>btnComputeTotal</td>
</tr>
<tr>
<td>label</td>
<td>lbl</td>
<td>lblInstructions</td>
</tr>
<tr>
<td>list box</td>
<td>lst</td>
<td>lstOutput</td>
</tr>
<tr>
<td>text box</td>
<td>txt</td>
<td>txtAddress</td>
</tr>
</tbody>
</table>

Fonts

- Proportional width fonts take up less space for "I" than for "W" – like Microsoft Sans Serif
- Fixed-width fonts take up the same amount of space for each character – like Courier New
- Fixed-width fonts are good for tables
Auto Hide

- Hides tool windows when not in use
- Vertical push pin icon indicates auto hide is disabled
- Click the push pin to make it horizontal and enable auto hide

Viewing the Code

- The GUI Forms Designer generates textual code
  - Prior to VB programmers wrote everything in textual code
- Click on the “Form1.VB” tab to see the code (not the design tab)
An Event Procedure Walkthrough

• An event is an action, such as:
  – The user clicks on a button
  – A form is minimized
  – The mouse enters or exits a control
  – The form is re-drawn

• Usually, nothing happens until an event occurs

The three steps in creating a VB.NET program:

1. Create the interface; that is, generate, position, and size the objects.
2. Set properties; that is, configure the appearance of the objects.
3. Write the code that executes when events occur.
Changing Properties

- Properties are changed in code with the following:
  \[ \text{controlName.property} = \text{setting} \]
- This is an assignment statement
- Examples:

  ```vbnet
  txtBox.ForeColor = Color.Red
  txtName.Text = "Hello There"
  txtName.Visible = False
  txtName.Location.X = 100
  ```

Adding Code to an Event

- To add code for an event:
  - In the VB Code Window select the control on the left side menu and the event of interest on the right side menu
  - Or double-click the control in the designer to bring up the most common event for that control

- Other methods for opening the Code window:
  - If the Code window is visible, click on it
  - Double-click anywhere on the Form window
  - Select the Code option from the View menu
  - Press the F7 method key anywhere on the design form
  - Select the View Code icon from the Project Window
Program Region

Event Procedures

Private Sub objectName_event(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles objectName.event

For now you can ignore most of this, aside from knowing the name of the subroutine:

Private Sub objectName_event(...) Handles objectName.event
Structure of an Event Procedure

Private Sub objectName_event(...)  
Handles objectName.event  
statements  ' Your code goes here  
End Sub

IntelliSense

Automatically pops up to give the programmer help.
Code for Walkthrough

Private Sub txtFirst_TextChanged(...) Handles txtFirst.TextChanged
    txtFirst.ForeColor = Color.Blue
End Sub

Private Sub btnRed_Click(...) Handles btnRed.Click
    txtFirst.ForeColor = Color.Red
End Sub

Private Sub txtFirst_Leave(...) Handles txtFirst.Leave
    txtFirst.ForeColor = Color.Black
End Sub

Assigning properties in code

• The following won't work:
  Form1.Text = "Demonstration"

• The current form is referred to by the keyword Me.
  Me.Text = "Demonstration"
The Declaration Statement of an Event Procedure

- A declaration statement for an event procedure:
  ```vbnet
  Private Sub btnOne_Click(...) Handles btnOne.Click
  ```

- The name can be changed at will. For example
  ```vbnet
  Private Sub ButtonPushed(...) Handles btnOne.Click
  ```

- Handling more than one event:
  ```vbnet
  Private Sub ButtonPushed(...) Handles btnOne.Click, btnTwo.Click
  ```

The MessageBox.Show Method

- The MessageBox.Show method is used to display a box with a message for the user
- The message box also contains a title and an icon
- General forms of the MessageBox.Show method
  - MessageBox.Show(text)
  - MessageBox.Show(text, caption)
  - MessageBox.Show(text, caption, buttons)
  - MessageBox.Show(text, caption, buttons, icon)
  - MessageBox.Show(text, caption, buttons, icon, defaultbutton)

- To do: Add a MessageBox.Show to the button click event
Console.WriteLine

- Another handy way to output information is to the Console:

  - Console.WriteLine("Hello there")
    • Outputs the message in double quotes and adds a newline
  - Console.Write("Hello again. ")
    • Outputs the message in double quotes without a newline

- Useful for debugging, don’t have to push the OK button and clutter up the screen with message boxes

Adding Additional Event Procedures

- Comments
  - Explanatory remarks made within a program
  - Indicated by an apostrophe or the keyword Rem

- Statement categories
  - An executable statement causes some specific action to be performed by the compiler or interpreter
  - A nonexecutable statement is a statement that describes some feature of either the program or its data but does not cause the computer to perform any action
Focus on Program Design and Implementation: Creating a Menu of Forms

• A main menu form can be displayed as an application’s opening window to provide the user with a summary of what the application can do

• The main menu form is created as either:
  – A set of buttons
  – A menu bar

Focus on Program Design and Implementation: Creating a Main Menu (Continued)

Figure 2-55: A Sample Main Menu Screen
Adding a Second Form

- From the Project menu, select Add New Item, and then Windows Form
  - Default name is “Form2.vb”
- To display a form:
  Dim varName As New FormName
  varName.Show()

  - E.g.:
    Dim secondForm as New Form2
    secondForm.Show()
- To make the current form invisible but retain its variables and components:
  Me.Hide() (could display again with Show())
- To destroy a form and unload its variables and components:
  Me.Close() (need to create a new one to display again)

Todo

- Implement program that can quit (End statement) and has a sub-form:
Knowing About: The Help Facility

- Visual Basic’s Help Facility can be accessed by selecting either the Contents, Search, or Index options from the Help menu.
- The Contents tab displays a Table of Contents for the documentation.
- The Index tab provides both a general index of topics and a text box for user entry of a specific topic.
- The Search tab provides a means of entering a search word or phrase.

Knowing About: The Help Facility (Continued)

- Dynamic Help
  - The Dynamic Help window displays a list of help topics that changes as you perform operations.
  - To open the Dynamic Help window, click Help on the menu bar and then click Dynamic Help.
- Context-sensitive Help
  - Context-sensitive Help immediately displays a relevant article for a topic.
  - To use this facility, select an object and press F1.
Common Programming Errors and Problems

• A common problem is not being able to locate all of the elements needed to create an application
  – Can usually get these windows back from the View menu
• A common error is forgetting to save a project at periodic intervals at design time

Summary

• Event-based programs execute program code depending on what events occur, which depends on what the user does
• GUIs are graphical user interfaces that provide the user with objects that recognize events
• The term “design time” refers to the period of time when a Visual Basic application is being developed under control of Visual Basic
• The term “run time” refers to the period of time when an application is executing
Summary

• A Visual Basic program consists of a visual part and a language part
• Basic steps for developing a Visual Basic program:
  – Create the GUI
  – Set the properties of each object on the interface
  – Write procedural code
• A form is used during design time to create a graphical user interface for a Visual Basic application
• The most commonly placed objects on a form are buttons, labels, and text boxes