2

Integrated Development Environment

Objectives

- To become familiar with the integrated development environment.
- To be able to create a standard executable.
- To be able to identify the controls in the toolbox.
- To be able to understand the types of commands contained in the menus and the tool bar.
- To be able to customize the form using properties.
- To be able to customize the form using controls.
- To be able to customize controls using properties.
- To be able to save a project.
- To be able to execute a simple program.
- To understand the difference between design mode and run mode.

Seeing is believing.
Proverb

Form ever follows function.
Louis Henri Sullivan

Intelligence . . . is the faculty of making artificial objects, especially tools to make tools.
Henri-Louis Bergson

Our life is frittered away by detail . . . Simplify, simplify.
Henry Thoreau

Outline

2.1 Introduction

Visual Basic's *Integrated Development Environment* (*IDE*) allows the programmer to create, run and debug Windows programs in one application (e.g., Visual Basic) without the need to open additional programs (i.e., a program to create the program, a program that executes the program, a program that debugs the program, etc.). In this chapter, we overview the Visual Basic IDE features and discuss how to create and execute a simple program.

2.2 Integrated Development Environment Overview

When Visual Basic is loaded, the ***New Project*** *dialog* shown in Fig. 2.1 is displayed. The **New Project** dialog allows the programmer to choose what type of Visual Basic program to create. **Standard EXE**, which is highlighted by default, allows the programmer to create a *standard executable* (i.e., a program that uses the most common Visual Basic features). We use **Standard EXE** for the majority of examples and exercises in this book, although later in the book you will learn about some of the other types.

Each type listed in Fig. 2.1 describes a group of related files called a *project*. Collectively, the project files form a Visual Basic program. The *project types* listed in Fig. 2.1 are the "Visual" in Visual Basic, because they contain predefined features for designing Windows programs. The programmer can use or leverage these existing project types to create powerful Windows applications in a fraction of the time it would normally take to create the same applications in other programming languages.

The **New Project** dialog contains three tabs—***New*** for creating a new project, ***Existing*** for opening an existing project and ***Recent*** for opening a project that has been previously loaded into the IDE. Note that the **New Project** dialog is displayed every time Visual Basic is executed unless the ***Don't show this dialog in the future*** *checkbox* (in the lower-left portion of Fig. 2.1) is checked. The number and names of the types appearing in the window can differ depending on the version of Visual Basic. Figure 2.1 shows *Visual Basic Enterprise Edition* project types (you may be working with another version of Visual Basic 6 that shows fewer project types).

A project type is opened by either double-clicking its icon with the left mouse button or by single-clicking the icon with the left mouse button and pressing ***Open***. Opening a project type closes the **New Project** dialog and loads the features associated with the selected project type into the IDE.

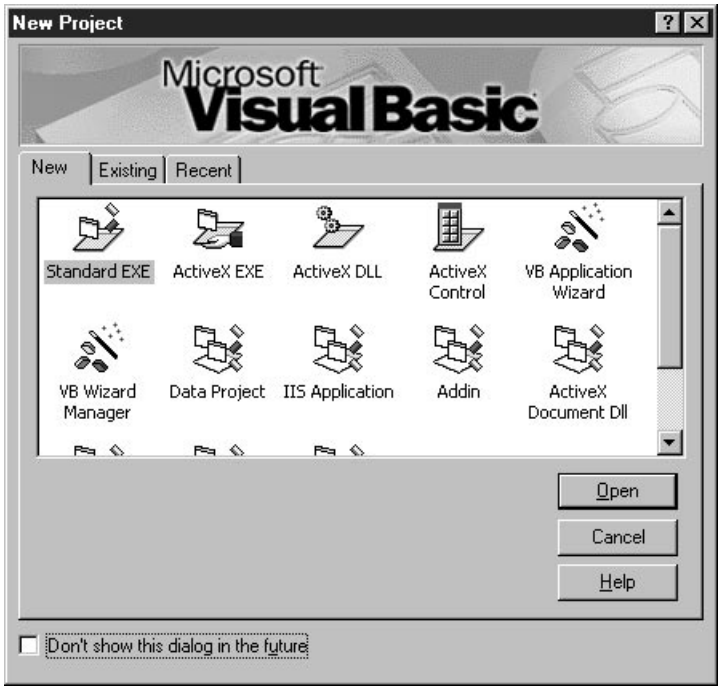


Fig. 2.1 **New Project** dialog.

Pressing ***Cancel*** closes the **New Project** dialog without opening a project type. Pressing **Help** opens the on-line assistance. We refer to single-clicking with the left mouse button as *selecting* or *clicking*, and we refer to double-clicking with the left mouse button simply as *double-clicking*.

Figure 2.2 shows the IDE after **Standard EXE** is selected. The top of the IDE window (the *title bar*) displays **Project1 - Microsoft Visual Basic [design]**. The environment consists of various windows, a *menu bar* and a *tool bar*. The menu bar contains several menus (**File**, **Edit**, **View**, etc.), each of which we overview shortly. The tool bar contains several icons that provide quick access to commonly used features. We discuss several of these tool bar icons in this chapter and others later in the book.

A **Standard EXE** project contains the following windows:

- **Project1 - Form1** (**Form**)
- **Form Layout**
- **Properties - Form1**
- **Project - Project1**
- Toolbox

The **Project - Form1** ***(Form)*** window contains a *form* named **Form1**, which is where the program's *Graphical User Interface* (*GUI*) will be displayed. A GUI is the visual portion of the program (i.e., buttons, etc.)–this is where the user enters data (called *inputs*) to the program and where the program displays its results (called *outputs*) for the user to read. We refer to the **Form1** window simply as "the form."

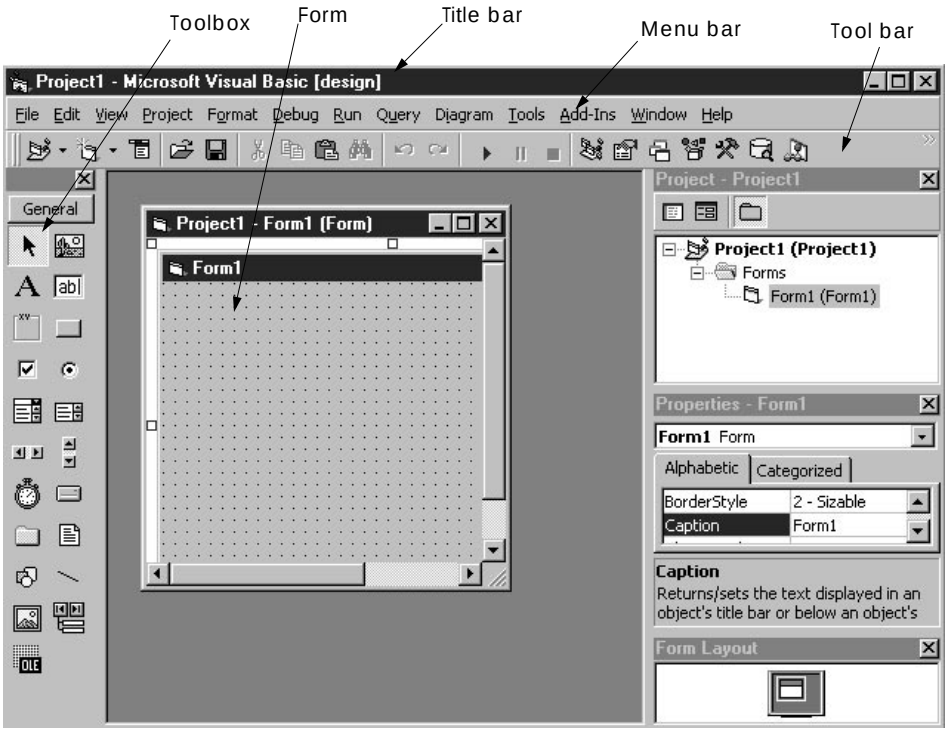


Fig. 2.2 IDE with a **Standard EXE** project open.

The **Form Layout** window enables the user to specify the form's position on the screen when the program is executed.

The **Properties - Form1** window displays form attributes or *properties* (i.e., color, font style, size, etc.). The **Project - Project1** window groups the project's files by type.

The toolbox contains *controls* for customizing the GUI (i.e., the form). Controls are GUI components such as buttons and checkboxes. We discuss a simple example at the end of this chapter that customizes a form with a control from the toolbox. We discuss toolbox controls throughout the book, especially in Chapters 10 and 11.

In the remainder of this chapter, we use these windows to create, manage and execute our first Visual Basic program.

2.3 Project Window

The window titled **Project - Project1** (Fig. 2.3) is called the ***Project Explorer*** and contains the project files. We refer to the **Project Explorer** window simply as the ***Project*** window.

The **Project** window's tool bar contains three *buttons*, namely ***View Code***, ***View Object*** and ***Toggle Folders***. When pressed, the **View Code** button displays a window for writing Visual Basic code. Writing code is the main subject of this book. **View Object**, when pressed, displays the form. Double-clicking **Form1 (Form1)** also displays the form. Both **View Code** and **View Object** are initially *disabled* (i.e., the buttons appear gray and

pressing them has no effect) unless **Form1 (Form1)** is selected (i.e., highlighted) as it is in Fig. 2.3. Figure 2.4 shows both the **View Code** and **View Object** buttons disabled. The **Toggle Folders** button *toggles* (i.e., alternately hides or shows) the ***Forms*** folder. When shown as in Fig. 2.3, the folder is visible, and when hidden as in Fig. 2.4, the folder is invisible. The **Forms** folder contains a listing of all forms in the current project. Early in the book our projects will have only one form.

Later in this chapter we will save projects and forms with more meaningful names. The current names **Project1**, **Form1**, etc. are default names provided by Visual Basic to help you get started. Visual Basic does many things automatically to minimize the amount of work you must do to create applications. In this regard, Visual Basic is the world's most popular *RAD (Rapid Applications Development)* programming language. The **Project** window becomes an important project management tool as projects become more complex (i.e., contain more forms and other support files).

2.4 Toolbox

The toolbox (Fig. 2.5) contains *controls* used to customize forms. Controls are prepackaged components that you reuse instead of writing them yourself–this helps you write programs faster. In this chapter, we overview the toolbox controls and in later chapters we discuss these controls in greater detail. Notice the box named **Data** displayed at the bottom of Fig. 2.5 when the *mouse pointer* (i.e., the **white arrow**) rests on the **Data** control. These boxed descriptions, called *tool tips*, are displayed by Visual Basic to tell you what each icon means. Tool tips are also displayed for many IDE features besides the toolbox. Figure 2.6 summarizes the toolbox controls.

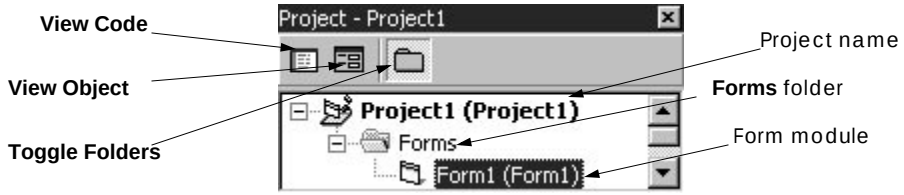


Fig. 2.3 **Project** window.

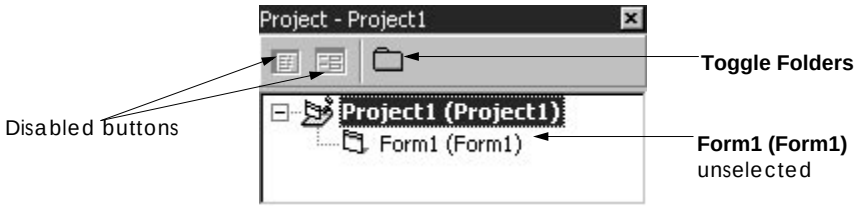


Fig. 2.4 **Project** window with disabled buttons and **Toggle Folders** set to off.

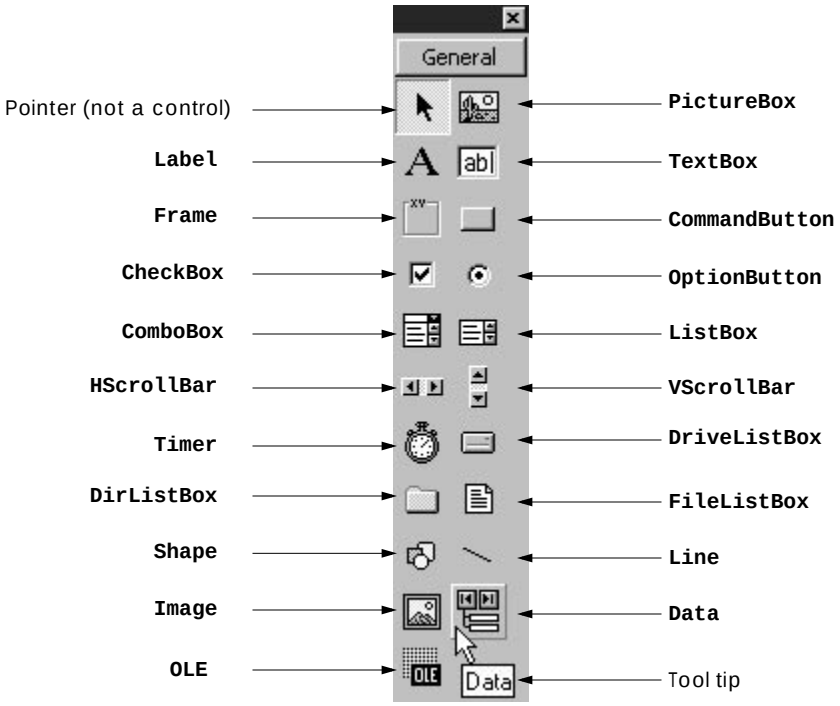


Fig. 2.5 **Toolbox** window.

Control	Description
Pointer	Used to interact with the controls on the form (i.e., resize them, move them, etc.). The pointer is not a control.
PictureBox	A control that displays images.
Label	A control that displays uneditable text to the user.
TextBox	A control for accepting user input. **TextBox**es can also display text.
Frame	A control for grouping other controls.
CommandButton	A control that represents a button. The user presses or clicks to initiate an action.
CheckBox	A control that provides the user with a toggle choice (checked or unchecked).
OptionButton	A "radio button." **OptionButtons** are used in groups where only one at a time can be **True** (i.e., on)—like the buttons on a car radio.
ListBox	A control that provides a list of items.
ComboBox	A control that provides a short list of items.

Fig. 2.6 Toolbox control summary (part 1 of 2).

Control	Description
HScrollBar	A horizontal scrollbar.
VScrollBar	A vertical scrollbar.
Timer	A control that performs a task at programmer-specified intervals. A **Timer** is not visible to the user.
DriveListBox	A control for accessing the system disk drives (**C:**, **A:**, etc.).
DirListBox	A control for accessing directories on a system.
FileListBox	A control for accessing files in a directory.
Shape	A control for drawing circles, rectangles, squares or ellipses.
Line	A control for drawing lines.
Image	A control for displaying images. The **Image** control does not provide as many capabilities as a **PictureBox**.
Data	A control for connecting to a database.
OLE	A control for interacting with other window applications.

Fig. 2.6 Toolbox control summary (part 2 of 2).

2.5 Form Layout Window

The **Form Layout** window (Fig. 2.7) specifies a form's position on the screen at runtime. The **Form Layout** window consists of an image representing the screen and the form's relative position on the screen. With the mouse pointer positioned over the form image, *drag* (i.e., hold down the left mouse button, then move the mouse and release the button) the form to a new location. Note that the mouse pointer changes shape when over the image representing the form. Later in the book we discuss the various shapes that the mouse pointer can assume.

2.6 Properties Window

The **Properties** window (Fig. 2.8) displays the *properties* for a form or control. Properties are attributes such as size, position, etc. Like a form, each control type has its own set of properties. Some properties, like ***Width*** and ***Height***, such as, are common to both forms and controls, while other properties are unique to a form or control. Controls often differ in the number and type of properties.

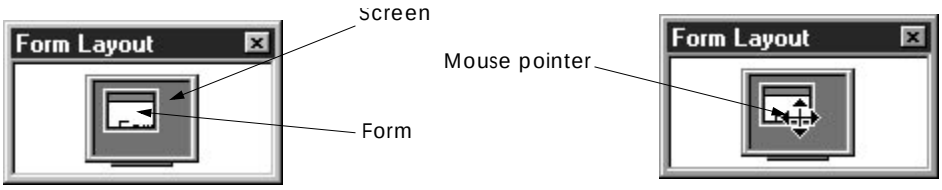


Fig. 2.7 **Form Layout** window.

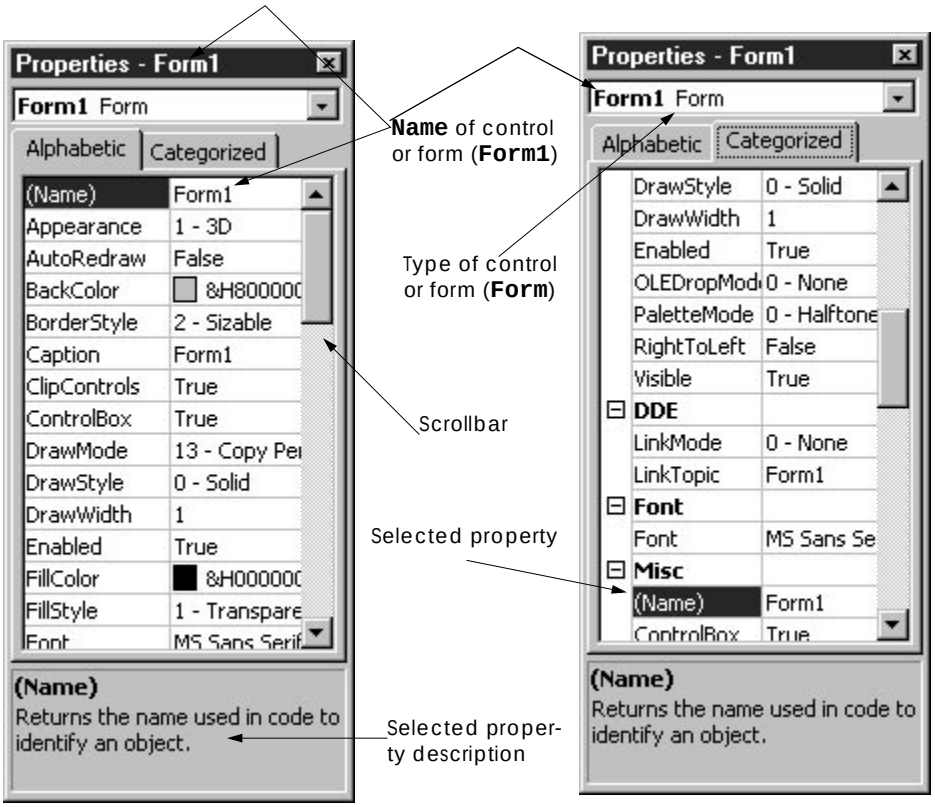


Fig. 2.8 **Properties** window with the **Alphabetic** and **Categorized** tabs.

Properties are listed either alphabetically (by selecting the ***Alphabetic*** tab) or categorically (by selecting the ***Categorized*** tab). **Alphabetic** lists the properties in alphabetical order and is the default. Clicking the **Categorized** tab lists properties by categories, such as **Appearance**, **Behavior**, **DDE**, **Font**, **Misc**, etc. The scrollbar can be used to scroll through the list of properties (by dragging the scrollbar up or down). We discuss setting individual properties later in this chapter and throughout the book.

2.7 Menu Bar and Tool Bar

Commands for developing, maintaining and executing programs are contained in the IDE's menus. Figure 2.9 shows the menus displayed on the *menu bar* for a **Standard EXE** project. Menus contain groups of related capabilities from which the user may select appropriate choices. The menus of Fig. 2.9 are summarized in Fig. 2.10. Note: Your version of Visual Basic may not have some of these menus.

Fig. 2.9 IDE menu bar.

Menu	Description
File	Contains options for opening projects, closing projects, printing projects, etc.
Edit	Contains options such as cut, paste, find, undo, delete, etc.
View	Contains options for displaying IDE windows and tool bars.
Project	Contains options for adding features such as forms to the project.
Format	Contains options for aligning and locking a form's controls.
Debug	Contains options for debugging.
Run	Contains options for executing a program, stopping a program, etc.
Query	Contains options for manipulating data retrieved from a database.
Diagram	Contains options for editing and viewing the design of databases.
Tools	Contains options for IDE tools and options for customizing the environment.
Add-Ins	Contains options for using, installing and removing add-ins. Add-ins are typically independent software vendor (ISV) products that extend Visual Basic's features.
Windows	Contains options for arranging and displaying windows.
Help	Contains options for getting help.

Fig. 2.10 Menu summary.

Rather than having to navigate the menus for certain commonly used commands, the programmer can select them from the *tool bar* (Fig. 2.11). The tool bar is comprised of pictures called *icons* that represent commands. Figure 2.11 shows the *standard tool bar* (i.e., the default tool bar). The figure indicates which menus contain the equivalent commands and it shows a few specific icons related to displaying the IDE windows.

2.8 A Simple Program: Displaying a Line of Text

In this section, we will create a program that displays the text "**`Welcome to Visual Basic!`**" on the form. The program consists of one form and uses one **`Label`** control to display the text. The program is a **Standard EXE**. We do not write a single line of program code. Instead, we introduce the techniques of *visual programming in which through various programmer gestures (such as using the mouse for pointing, clicking, dragging and*

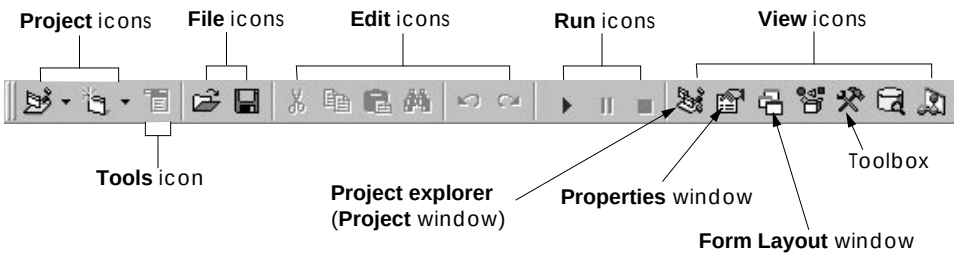


Fig. 2.11 Tool bar icons.

dropping) we provide Visual Basic with sufficient information so that it can automatically generate all or a major portion of the program code for our program. Figure 2.12 shows the program at runtime. In the next chapter, we begin our discussion of writing program code. Throughout the book we will produce increasingly substantial and powerful programs. Visual Basic programming involves a combination of writing a portion of the program code yourself and having Visual Basic generate the remaining code automatically.

Here are the steps you perform to create, run and terminate this first program:

1. *Setting the form's title bar*. The form's ***`Caption`*** *property* determines what is displayed in the form's title bar. In the **Properties** window, set the **`Caption`** property to **`Fig. 2.12: A Simple Program`**. To change the value of this property, click in the field next to **`Caption`** (this field displays **`Form1`**). Delete the existing value using the *Backspace* key or *Delete* key and enter the new value. Hit the *Enter* key (*Return* key). Note: As you enter a new value for the **`Caption`** property the form's title bar changes in response to what you are typing.
2. *Setting the form's* ***`Name`*** *property*. The ***`Name`*** *property* identifies a form or control. Set the **`Name`** property to **`frmFig02_12`**. After the property is set, note the changes to the **Properties** window and **Project** window as shown in Fig. 2.13.

Fig. 2.12 Program at run-time.

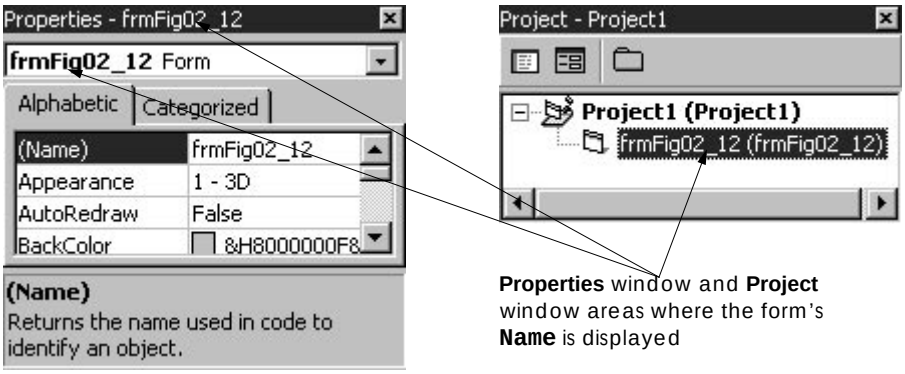


Fig. 2.13 **Properties** window and **Project** window after the **`Name`** property is set.

Good Programming Practice 2.1

Prefix the **`Name`** *of each form with* **`frm`** *to make form objects easy to identify.*

3. *Resizing the form*. Click and drag one of the form's *enabled sizing handles* (the small squares around the form shown in Fig. 2.14). White sizing handles are *disabled* and the programmer cannot use them to resize the form. Sizing handles that are black are enabled and can be used for resizing. Size the form according to your own preference. Sizing handles are not visible during program execution.
4. *Centering the form*. Center the form using the **Form Layout** window (Fig. 2.15). This causes the form to display in the center of the monitor when the program is executed.

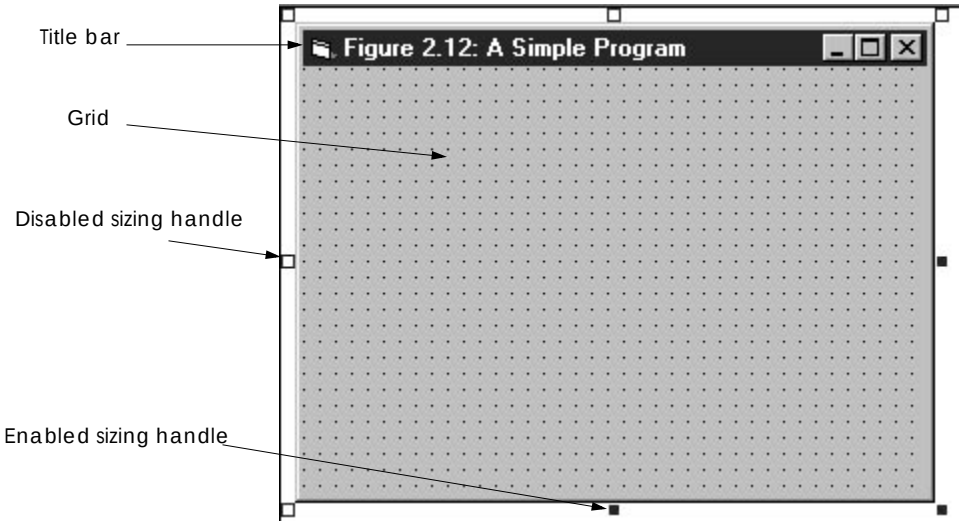


Fig. 2.14 Form with sizing handles.

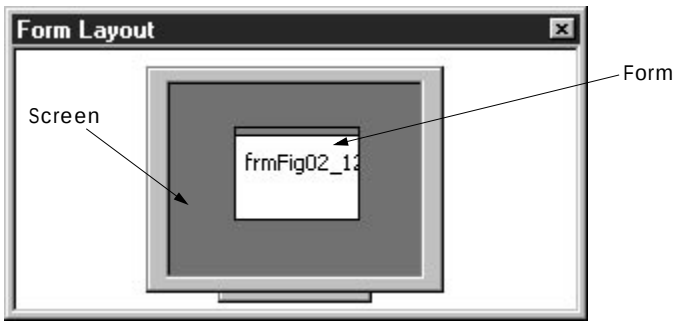


Fig. 2.15 **Form Layout** window with form centered.

5. *Changing the form's background color*. The ***BackColor*** *property* specifies a form or control's background color. Clicking **BackColor** in the **Properties** window causes a *down-arrow* button to appear next to the property value as shown in Fig. 2.16. When clicked, the down arrow displays a window with the tabs ***System*** (the default) and ***Palette***. Click the **Palette** tab to display the *palette* (a group of colors from which the user selects one by clicking). Select the box representing yellow. The palette disappears and the form's background color changes to yellow. Note that **BackColor** displays a small rectangle representing the current color.
6. *Adding a* ***Label*** *control to the form*. Double-click the toolbox's **Label** control to create a **Label** with sizing handles in the center of the form (Fig. 2.17). The **Label** displays the word **Label1** by default. Double clicking any toolbox control results in a control being created and placed in the center of the form. When the sizing handles appear around the **Label**, the **Properties** window displays the **Label**'s properties. Clicking the form causes the form's properties to be displayed in the **Properties** window.
7. *Setting the* ***Label****'s display*. The **Label**'s **Caption** property determines what text (if any) the **Label** displays. The form and **Label** each have their own **Caption** property—with each being completely independent of the other. Forms and controls can have properties with the same name without conflict. Set the **Label**'s **Caption** property to **Welcome to Visual Basic!** (Fig. 2.18). The **Label** displays each character as it is typed. Note that when the edge of the **Label** is reached, the text automatically wraps to the next line.

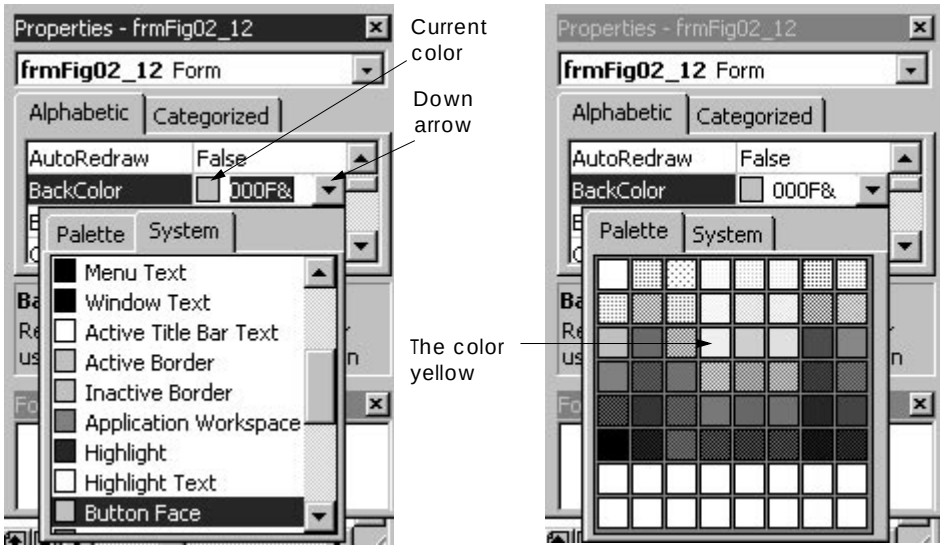


Fig. 2.16 Changing the **BackColor**.

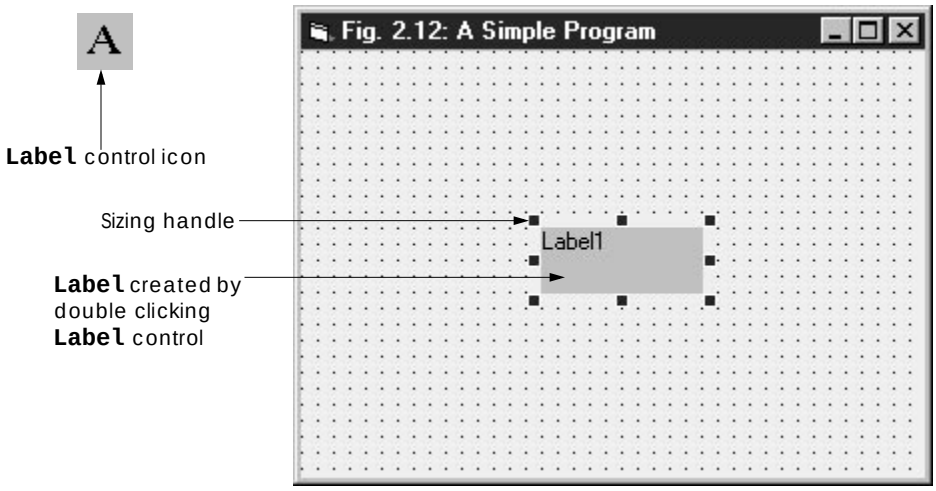


Fig. 2.17 A `Label` placed on the form.

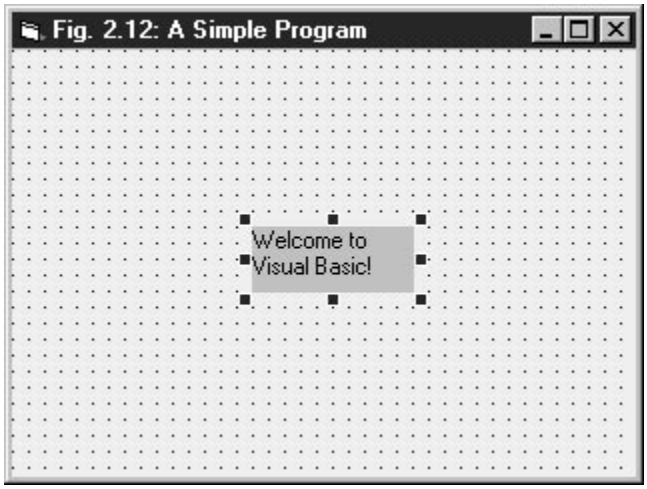


Fig. 2.18 The `Caption` property set for the `Label`.

8. *Naming the* ***`Label`***. The `Label`'s `Name` property is used to identify the `Label`. The default name for the `Label` we just created is `Label1`. Set the `Name` property to `lblWelcome`.

Good Programming Practice 2.2

Prefix the ***`Name`*** *of each* ***`Label`*** *with* ***`lbl`*** *to make* ***`Label`*** *objects easy to identify.*

Good Programming Practice 2.3

Following widely accepted naming conventions can make your program clearer—especially to other people.

9. *Customizing the* **`Label`***'s colors*. Like a form, a control's **`BackColor`** is gray by default, and we wish to change it to yellow. There are two ways to accomplish this using the **`Label`**'s properties. One way is to change the **`Label`**'s **`BackColor`** property to yellow—which works well as long as the form's **`BackColor`** does not change. If the form's **`BackColor`** changes, the **`Label BackColor`** remains yellow. The second way is to change the **`Label`**'s ***`BackStyle`*** property from ***`Opaque`*** (i.e., solid) to ***`Transparent`*** (i.e., see through). The **`Label`**'s ***`ForeColor`*** *property* determines the color in which text is displayed. Set the **`ForeColor`** to blue using the techniques discussed in Step 5.
10. *Setting the* **`Label`***'s font size and aligning the* **`Label`***'s text*. Clicking the ***`Font`*** *property* value causes an *ellipsis* button to appear (Fig. 2.20). When this button is pressed, the ***`Font`*** window of Fig. 2.21 appears. The font name (**MS Sans Serif**, **Serif**, etc.), font style (**Regular**, **Bold**, etc.) and font size (**8**, **10**, etc.) can be selected. The current font is applied to the text in the **Sample** area. Under the **Size** category select **24** and press **OK**. Next, select the ***`Alignment`*** *property*—which determines how the text is aligned within the **`Label`** boundaries. The three **`Alignment`** choices are ***`Left Justify`*** (the default), ***`Right Justify`*** and ***`Center`***. Select **`Center`**. At this point, you might notice that the **`Label`** size is too small for the font size. In the next step we will resize the **`Label`**.
11. *Positioning and resizing the* **`Label`**. Resize the **`Label`** using the **`Label`**'s sizing handles, such that the **`Label`** appears similar to that of Fig. 2.22. Note the change in the mouse pointer when it is placed over a sizing handle. Center the **`Label`** on the form by dragging the **`Label`**. The *grid* dots on the background of the form are used for aligning controls and are only visible at design time.

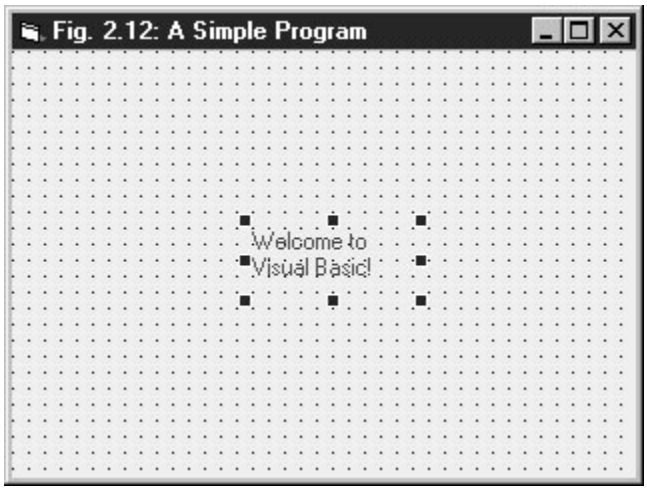


Fig. 2.19 Transparent **`Label`** with yellow **`ForeColor`**.

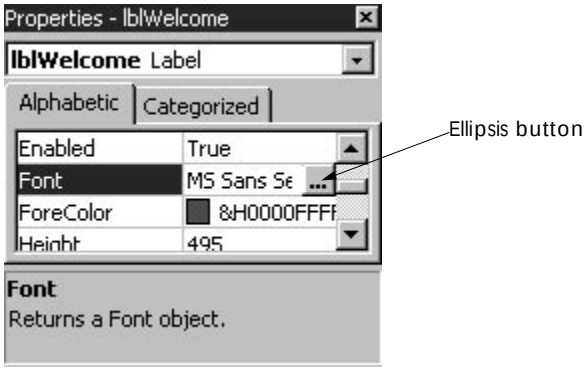


Fig. 2.20 **Properties** window displaying the **Label**'s properties.

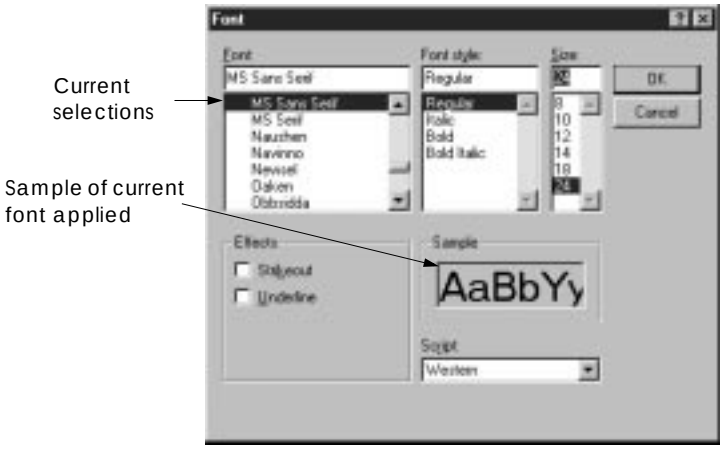


Fig. 2.21 **Font** window for selecting fonts, styles and sizes.

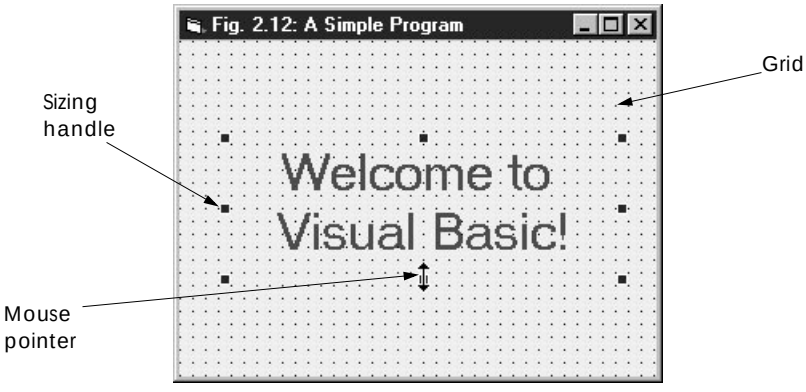


Fig. 2.22 Resizing the **Label**.

12. *Saving the project*. Click the ***Save Project*** *tool bar icon* (Fig. 2.23) or select ***Save Project*** / ***Save Project As...*** from the **File** menu to display the ***Save File As...*** dialog (Fig. 2.23). The **Save File As...** dialog specifies the *form file name* that will store all the form's information (i.e., properties, etc.). We save our file in the **c:\books\vbhtp\examples\chap02\fig02_12** directory. You are free to choose whatever directory you want. The window provides the capabilities to visually navigate the directories and to create new folders. After specifying the name and directory, click the ***Save*** button.

The next dialog that appears is the ***Save Project As...*** dialog. The features of this dialog are identical to the features of the **Save File As...** dialog, except that now we specify the *project file name*. The project file stores the name and location of every file in the project. We save the project in the same directory as the form (**c:\books\vbhtp\examples\chap02\fig02_12**). Again you are free to save the project file in any directory you choose. Figure 2.25 shows the **Project** window after the project is saved.

13. *Running the program*. Prior to this step, we have been working in the IDE *design mode* (i.e., the program is not executing). While in design mode, the programmer has access to all the environment windows (i.e., toolbox and **Properties**), menus, tool bars, etc. While in *run mode* the program is executing and the user can only interact with a few IDE features. Features that are not available are disabled. To execute or run your program, click the ***Start*** button or select **Start** from the **Run** menu. Figure 2.26 shows the IDE in run mode. Note that the IDE title bar displays **[Run]** and that most tool bar icons are disabled. Also note that the ***Immediate*** window appears at runtime. The **Immediate** window is primarily used for debugging (i.e., removing errors from your program) and is discussed in Chapter 13.

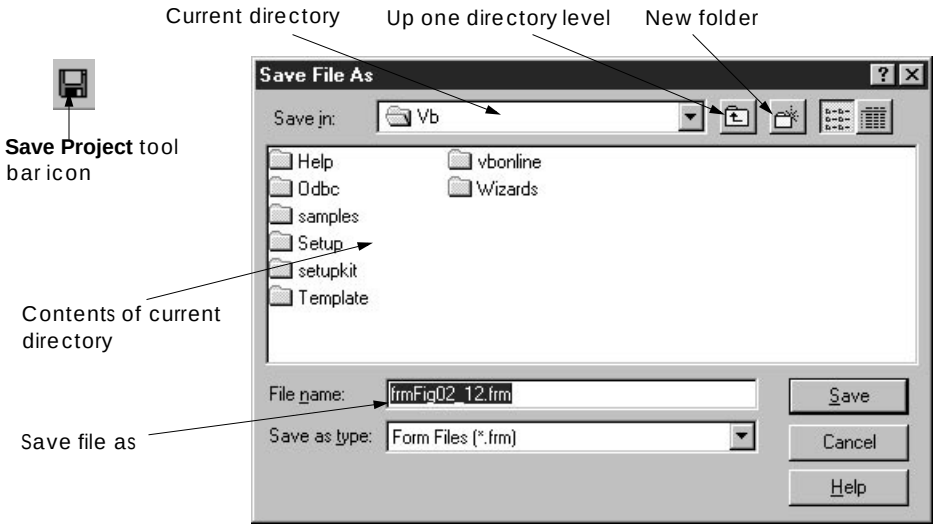


Fig. 2.23 Save File As... dialog.

Fig. 2.24 **Save Project As...** dialog.

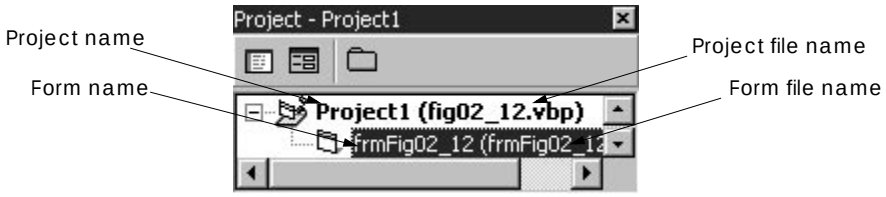


Fig. 2.25 **Project** window after saving project.

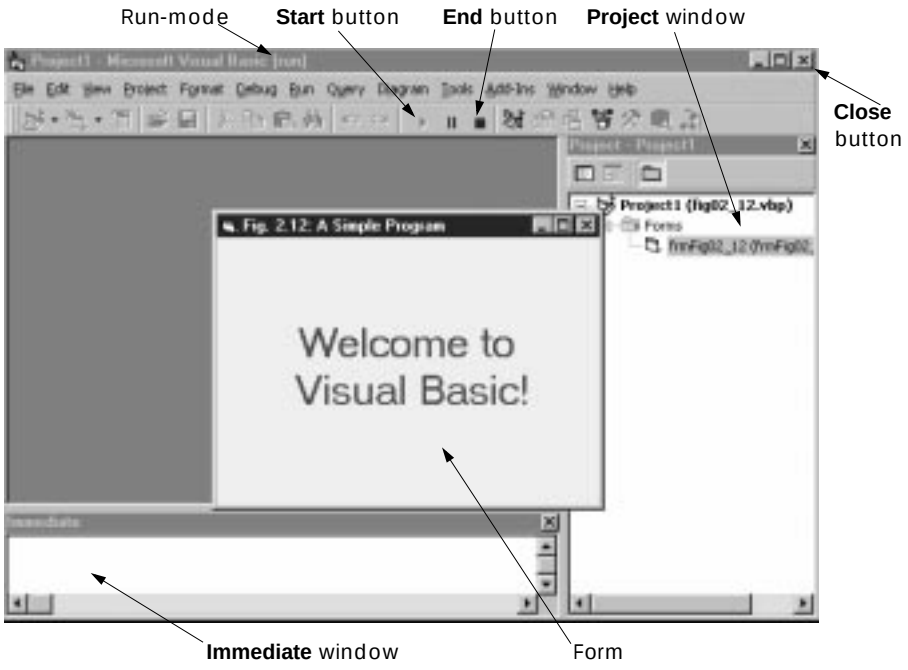


Fig. 2.26 IDE during execution.

14. *Terminating execution*. Clicking form's ***Close*** button icon (i.e., the "X" at the top right corner of Fig. 2.26) or by clicking the tool bar's ***End*** button terminates program execution and places the IDE in design mode.

Summary

- The Visual Basic Integrated Development Environment (IDE) allows the programmer to create, run and debug Windows programs.
- The **New Project** dialog allows the programmer to choose what type of Visual Basic program to create. **Standard EXE** allows the programmer to create a standard executable (i.e., a program that uses the most common Visual Basic features).
- The **New Project** dialog contains three tabs—**New** for creating a new project, **Existing** for opening an existing project and **Recent** for opening a project that has been previously loaded into the IDE.
- We refer to single clicking with the left mouse button as selecting or clicking, and we refer to double-clicking with the left mouse button simply as double-clicking.
- A **Standard EXE** project contains the following windows: **Project1 - Form1** (**Form**), **Form Layout, Properties - Form1, Project - Project1** and the toolbox.
- The **Project - Form1 (Form)** window contains a child window named **Form1**, which is where the program's Graphical User Interface (GUI) will be displayed. A GUI is the visual portion of the program (i.e., buttons, checkboxes, etc.)–this is where the user enters data (called inputs) to the program and where the program displays its results (called outputs) for the user to read.
- The **Form Layout** window enables the user to specify the form's position on the screen when the program is executed.
- The **Properties - Form1** window displays form attributes or properties (i.e., color, font style, size, etc.).
- The toolbox contains controls for customizing the GUI (i.e., the form). Controls are GUI components such as buttons and checkboxes.
- The window titled **Project - Project1** (Fig. 2.3) is called the **Project Explorer** and contains the project files. We refer to the **Project Explorer** window simply as the **Project** window.
- The **Project** window's tool bar contains three buttons, namely **View Code**, **View Object** and **Toggle Folders**. When pressed, the **View Code** button displays a window for writing Visual Basic code.
- **View Object**, when pressed, displays the form. Double-clicking **Form1 (Form1)** also displays the form.
- Both **View Code** and **View Object** are initially disabled (i.e., the buttons appear gray and pressing them has no affect) unless **Form1 (Form1)** is selected (i.e., highlighted).
- The **Toggle Folders** button toggles (i.e., alternately hides or shows) the **Forms** folder.
- The **Forms** folder contains a listing of all forms in the current project.
- Visual Basic does many things automatically to minimize the amount of work you must do to create applications. In this regard, Visual Basic is the world's most popular RAD (Rapid Applications Development) programming language.
- The **Project** window becomes an important project management tool as projects become more complex (i.e., contain more forms and other support files).
- The toolbox contains controls used to customize forms. Controls are prepackaged components that you reuse instead of writing them yourself–this helps you write programs faster.

- Tool tips are displayed by Visual Basic to tell you what each icon means.
- The pointer is used to interact with the controls on the form (i.e., resize them, move them, etc.). The pointer is not a control.
- A **`PictureBox`** is a control that displays images.
- A **`Label`** is a control that displays uneditable text to the user.
- A **`TextBox`** is a control for accepting user input. **`TextBox`**es can also display text.
- A **`Frame`** is a control for grouping other controls.
- A **`CheckBox`** is a control that provides the user with a toggle choice (checked or unchecked).
- An **`OptionButton`** is a "radio button." **`OptionButtons`** are used in groups where only one at a time can be **`True`** (i.e., on), just like the buttons on a car radio.
- **`ListBox`** is a control that provides a list of items.
- A **`ComboBox`** is a control that provides a short list of items.
- An **`HScrollBar`** is a horizontal scrollbar.
- A **`VScrollBar`** is a vertical scrollbar.
- A **`Timer`** is a control that performs a task at programmer-specified intervals. A **`Timer`** is not visible to the user.
- A **`DriveListBox`** is a control for accessing the system disk drives (**`C:`**, **`A:`**, etc.).
- A **`DirListBox`** is a control for accessing directories on a system.
- A **`FileListBox`** is a control for accessing files in a directory.
- A **`Shape`** is a control for drawing circles, rectangles, squares or ellipses.
- A **`Line`** is a control for drawing lines.
- An **`Image`** is a control for displaying images. The **`Image`** control does not provide as many capabilities as a **`PictureBox`**.
- A **`Data`** control provides a means for connecting to a database.
- An **`OLE`** control for interacting with other Windows applications.
- The **Form Layout** window specifies a form's position on the screen at run-time. The **Form Layout** window consists of an image representing the screen and the form's relative position on the screen. To reposition the form on the screen, position the mouse pointer over the form image, then drag (i.e., hold down the left mouse button, then move the mouse and release the button) the form to a new location.
- The **Properties** window displays the properties for a forms and controls.
- Properties are attributes such as size, position, etc. Like a form, each control type has its own set of properties. Some properties, such as, **`Width`** and **`Height`**, are common to both forms and controls, while other properties are unique to a form or control. Controls often differ in the number and type of properties.
- Properties are listed either alphabetically (by selecting the **Alphabetic** tab) or categorically (by selecting the **Categorized** tab). **Alphabetic** lists the properties in alphabetical order and is the default. Clicking the **Categorized** tab lists properties by categories, such as **Appearance**, **Behavior**, **DDE**, **Font**, **Misc**, etc. The scrollbar can be used to scroll through the list of properties (by dragging the scrollbar up or down).
- Commands for developing, maintaining and executing programs are contained in the IDE's menus. Menus contain groups of related capabilities from which the user may select appropriate choices.

- The **File** menu contains options for opening projects, closing projects, printing projects, etc.
- The **Edit** menu contains options such as cut, paste, find, undo, delete, etc.
- The **View** menu contains options for displaying IDE windows and tool bars.
- The **Project** menu contains options for adding features such as forms to the project.
- The **Format** menu contains options for aligning and locking a form's controls.
- The **Debug** menu contains options for debugging.
- The **Run** menu contains options for executing a program, stopping a program, etc.
- The **Query** menu contains options for manipulating data retrieved from a database.
- The **Diagram** menu contains options for editing and viewing the design of databases.
- The **Tools** menu contains options for IDE tools and options for customizing the environment.
- The **Add-Ins** menu contains options for using, installing and removing add-ins. Add-ins are typically independent software vendor (ISV) products that extend the features of Visual Basic.
- The **Windows** menu contains options for arranging and displaying windows.
- The **Help** menu contains options for getting help.
- Rather navigating the menus for certain commonly used commands, the programmer can select them from the tool bar. The tool bar is comprised of pictures called icons that represent commands.
- We do not write a single line of code—instead, we introduce the technique of visual programming.
- The form's **`Caption`** property determines what is displayed in the form's title bar.
- To change the value of the **`Caption`** property, click in the field next to **`Caption`**. Delete the existing value using the Backspace key or Delete key and enter the new value. Hit the Enter key (Return key). As you enter a new value for the **`Caption`** property the form's title bar changes in response to what you are typing.
- The **`Name`** property identifies a form or control.
- To resize a form, click and drag one of the form's enabled sizing handles (the small squares around the form). White sizing handles are disabled and the programmer cannot use them to resize the form. Sizing handles that are black are enabled and can be used for resizing. Sizing handles are not visible during program execution.
- A form can be centered by using the **Form Layout** window.
- The **`BackColor`** property specifies a form or control's background color. Clicking **`BackColor`** in the **Properties** window causes a down-arrow button to appear next to the property value. When clicked, the down arrow displays a window with the tabs **System** (the default) and **Palette**. Click the **Palette** tab to display the palette (a group of colors from which the user selects one by clicking). The **`BackColor`** displays a small rectangle representing the current color.
- To add a **`Label`** control to a form, double-click the toolbox's **`Label`** control to create a **`Label`** with sizing handles in the center of the form. The **`Label`** displays the word **`Label1`** by default. Double-clicking any toolbox control results in a control being created and placed in the center of the form. When the sizing handles appear around the **`Label`**, the **Properties** window displays the **`Label`** properties. Clicking the form causes the form's properties to be displayed in the **Properties** window.
- The **`Label`**'s **`Caption`** property determines what text (if any) the **`Label`** displays. The form and **`Label`** each have their own **`Caption`** property—with each being completely independent of the other. Forms and controls can have properties with the same name without conflict. The **`Label`** displays each character as it is typed. Note that when the edge of the **`Label`** is reached, the text automatically wraps to the next line.

- A control's **BackColor** is gray by default.
- A **Label** has a **BackColor** property. A **Label**'s **BackStyle** property can be **Opaque** (i.e., solid) or **Transparent** (i.e., see through).
- A **Label**'s **ForeColor** property determines the color in which text is displayed.
- Clicking a **Label**'s **Font** property value causes an ellipsis button to appear. When this button is pressed, the **Font** dialog appears. The font name (**MS Sans Serif**, **Serif**, etc.), font style (**Regular**, **Bold**, etc.) and font size (**8**, **10**, etc.) can be selected. The current font is applied to the text in the **Sample** area.
- A **Label**'s **Alignment** property determines how text is aligned within the **Label** boundaries. The three **Alignment** choices are **Left Justify** (the default), **Right Justify** and **Center**.
- A **Label** can be resized using its sizing handles. The shape of the mouse pointer changes when it is placed over a sizing handle. The grid dots on the background of the form are used for aligning controls at design time.
- Click the **Save Project** tool bar icon or select **Save Project** / **Save Project As...** from the **File** menu to display the **Save File As...** dialog. The **Save File As...** dialog specifies the form file name that will store all the form's information (i.e., properties, etc.) as well as its directory location.
- The **Save Project As...** dialog's features are identical to those of the **Save File As...** dialog, except that you use it to specify the project file name. The project file stores the name and location of every file in the project.
- In IDE design mode, the program is not executing. While in design mode, the programmer has access to all the environment windows (i.e.f, toolbox and **Properties**, menus, tool bars, etc.).
- While in run mode the program is executing and the user can interact with only a few IDE features. Features that are not available are disabled.
- To execute or run a program, click the **Start** button or select **Start** from the **Run** menu. Note that the IDE title bar displays **[Run]** and that most tool bar icons are disabled. Also note that the **Immediate** window appears at run-time. The **Immediate** window is primarily used for debugging.
- Clicking form's **Close** button icon (i.e., the "X" at the form's top-right corner or clicking the tool bar's **End** button terminates program execution and places the IDE in design mode.

Terminology

active window
Add-Ins menu
Alignment property of a **Label**
Alignment property's **Left Justify** value
Alignment property's **Right Justify** value
Alphabetic tab of **Properties** window
BackColor property of a **Form**
Backspace key
BackStyle property
button
Cancel button
Caption property
Categorized tab of a **Properties** window
Center value of **Alignment** property
checkbox
click and drag
clicking
Close button
controls
Debug menu
Delete key
design mode
design time
disabled button
disabled sizing handle
Don't show this dialog in the future
double-clicking
down-arrow button

dragging
Edit menu
ellipsis button
enabled sizing handle
End button
Enter key
Enterprise Edition
Existing tab of **`New Project`** dialog
File menu
focus
font
`Font` property of a **Label**
Font window
`ForeColor` property of a **Label**
form
Format menu
Form Layout window
`Forms` folder
graphical user interface (GUI)
grid
`Height` property
Help menu
icons
Immediate window
integrated development environment (IDE)
`Label` control
menu
menu bar
mouse pointer
`Name` property
New Project dialog
New tab of **New Project** dialog
`Opaque` value of **`Back Style`** property
Open button
palette of colors
Palette tab
project
Project Explorer window
Project menu
project types
Project window
properties
Properties window
RAD (Rapid Applications Development)
radio buttons
Recent tab of **New Project** dialog
Return key
Run menu
run mode
run-time
Sample area of **Font** window
Save button
Save File As... dialog
Save Project As... dialog
Save Project tool bar icon
selecting
sizing handle
Standard EXE project type
standard executable
standard tool bar
standard Visual Basic executable
Start button
System tab
title bar
toggle
Toggle Folders button in **Project** window
tool bar
toolbox
Toolbox window
Tools menu
tool tips
`Transparent` value
VB Application Wizard project type
VB Learning Edition Controls project type
View Code button in **Project** window
View menu
View Objects button in **Project** window
visual programming
`Width` property
Window menu

Good Programming Practices

2.1 Prefix the **`Name`** of each form with **`frm`** to make form objects easy to identify.

2.2 Prefix the **`Name`** of each **`Label`** with **`lbl`** to make **`Label`** objects easy to identify.

2.3 Following widely accepted naming conventions can make your program clearer—especially to other people.

Self-Review Exercises

2.1 Fill in the blanks in each of the following:

a) A _____ is a customizable window.

b) The form's _____ is used to visually align controls.
c) The form's _____ property specifies the text for the form's title bar.
d) The _____ window has a dark-colored title bar and is said to have the _____.
e) The _____ window determines where a form will appear on the screen at execution.
f) The _____ window contains the program files.

2.2 State whether each of the following is *true* or *false*. If *false*, explain why.
a) The tool bar contains the control icons.
b) The **Project** window is also called the **Project Explorer**.
c) The tool bar provides a convenient way to execute certain menu commands.
d) The **Properties** window is also called the **Immediate** window.
e) A form's sizing handles are always enabled.
f) The pointer is not a control.

2.3 Match the control name with the proper toolbox icon in Fig. 2.27. Note that **`OLE`** is not shown.

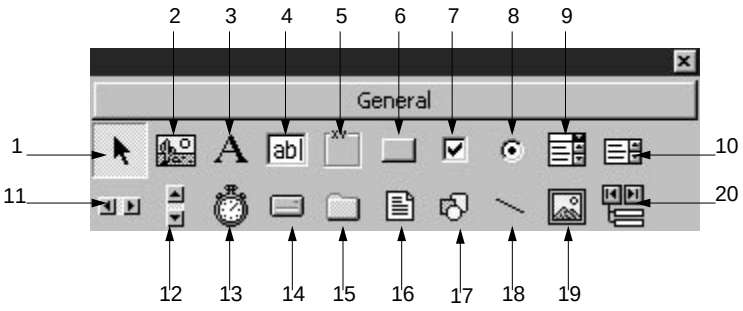


Fig. 2.27 Toolbox.

a) **`CommandButton`**
b) **`OptionButton`**
c) **`Label`**
d) **`Line`**
e) **`Frame`**
f) **`Image`**
g) **`PictureBox`**
h) **`VScrollBar`** (Vertical scrollbar)
i) Pointer
j) **`ComboBox`**
k) **`TextBox`**
l) **`HScrollBar`** (Horizontal scrollbar)
m) **`DriveListBox`**
n) **`Shape`**
o) **`CheckBox`**
p) **`Data`**
q) **`FileListBox`**
r) **`ListBox`**
s) **`Timer`**
t) **`DirListBox`** (Directory list box)

Answers to Self-Review Exercises

2.1 a) form. b) grid. c) **`Caption`**. d) active, focus. e) **Form Layout**. f) **Project** or **Project Explorer**.

2.2 a) False. The toolbox contains the control icons.
b) True.
c) True.
d) False. The **Immediate** window is a distinctly different window.
e) False. Some of the form's sizing handles are disabled.
f) True.

2.3 a) 6. b) 8. c) 3. d) 18. e) 5. f) 19. g) 2. h) 12. i) 1. j) 9. k) 4. l) 11. m) 14. n) 17. o) 7. p) 20. q) 16. r) 10. s) 13. t) 15.

Exercises

2.4 Fill in the blanks in each of the following statements:

a) The _____ contains a variety of colors, from which the programmer selects one.
b) The three values of the **`Alignment`** property are _____, _____ and _____.
c) The _____ property changes a control's foreground color.
d) IDE is an abbreviation for _____.
e) Clicking the _____ on the toolbar executes the program.
f) The _____ property identifies a form and is often prefixed with **`frm`**.
g) GUI is an abbreviation for _____.
h) A _____ is a group of related files.

2.5 State which of the following are *true* and which are *false*. If *false*, explain why.

a) At run-time, a form's grid is visible.
b) A tool tip identifies an IDE feature.
c) A **`Label`**'s **`Text`** property determines what text is displayed to the user.
d) At design-time, almost every IDE feature is available.
e) When placed over an enabled sizing handle, the mouse pointer changes.
f) A **`Label`** displays uneditable text to the user.
g) A form and **`Label`** have an identical set of properties.

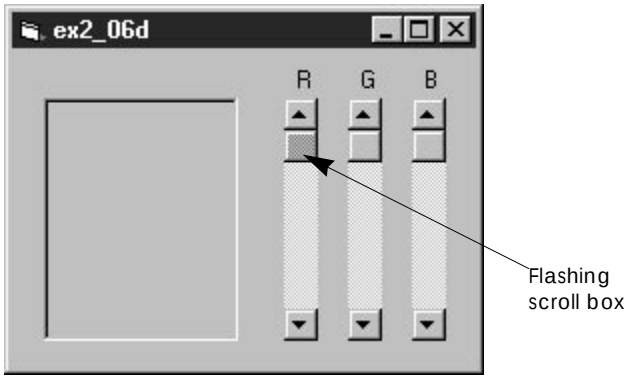
2.6 Build the following GUIs (you need not provide any functionality). Execute each program and determine what happens when a control is clicked with the mouse.

a) This GUI consists of three **`Label`**s colored yellow, red and black.

b) This GUI consist of one **`Label`** and eighteen **`CommandButtons`**. Note: You must modify the **`Label`**'s **`BorderStyle`** property. Also note that the dotted line around the six (**`6`**) button (it can be any of your buttons) appears during run mode.

c) The following GUI consists of one **Label**, one **CommandButton** and four **OptionButton**s. Note: The black dot in **Dog** automatically appears at run-time but may appear in a different one of your buttons.

d) The following GUI consists of three **VScrollBar**s and two **Label**s. Note: One **Label** requires its **BorderStyle** property changed. Also note that one **VScrollBar**'s *scroll box* automatically flashes at run-time.



2.7 Briefly describe each of the following IDE features:
a) tool bar
b) menu bar
c) toolbox
d) control
e) form
f) project
g) title bar

2.8 Briefly describe the differences between design mode and run mode.

2.9 Compare a form's properties to a **Label**'s properties. Make a list of all the properties that are common to both. Now, summarize only the properties on the list we have discussed in this chapter.

2.10 Why do you think that the toolbox, the form and the **Properties** window are crucial to the concept of visual programming?