Chapter 2

The Grand Tour

G O A L S

After working through this chapter, you will:

Recognize the components of the Visual Basic programming environment.

Know how to place textboxes, label controls, and command buttons on a Visual Basic form.

Be able to change the properties of controls.

Have explored how Visual Basic responds to the Click and KeyPress events.

Have written a program from beginning to end, including creating, saving, and printing a project as well as creating an EXE file.

OVERVIEW

Before you can program in Visual Basic, you need to understand the environment in which you are working and the tools at your disposal. It's like walking onto a basketball court for the first time. You have to understand the tools you will have, the process, and the expected end result. For a first basketball game, those would be a basketball (tool), working with teammates to make baskets (process), and having a good time (end result).

After you become comfortable with being in the Visual Basic environment, you will create your first program. You might ask how programming can be that easy. Here you are in the second chapter of the book, and you are programming.

Visual Basic itself is the reason. Not too long ago, you would have had to write a great deal of code to create even the simplest of programs. Now Visual Basic handles much of the code writing for you. You are left with the most challenging part of programming: thinking through programs so that they work well and are appealing to the people who run them.

Touring Visual Basic

This section covers the Visual Basic environment. You will open the application, then take a quick tour through the windows and the Toolbox that you see displayed on the screen.

Using online help is an important aspect of working with any Windows application. Often, you can find an answer to a question much faster by looking in online help than by paging through a user's manual. You need to become comfortable with looking in online help and finding information there.

Start Visual Basic for Windows now. If necessary, double-click on its program group icon in the Program Manager to open the group; then double-click on the Visual Basic icon (Figure 2-1).

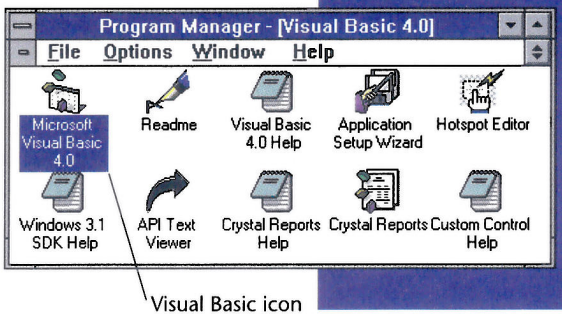


Figure 2-1
Selecting the Visual Basic icon

The Opening Screen

On the opening screen of Visual Basic are five elements:

- Menu bar and toolbar
- Window titled Form1
- Project window
- Toolbox
- Properties window

MENU BAR AND TOOLBAR

The menu bar and toolbar appear across the top of the screen (Figure 2-2). As in any Windows program, you use commands from the menu bar to perform tasks, such as opening a file. For some of the more common commands, you can also click on an icon in the toolbar.

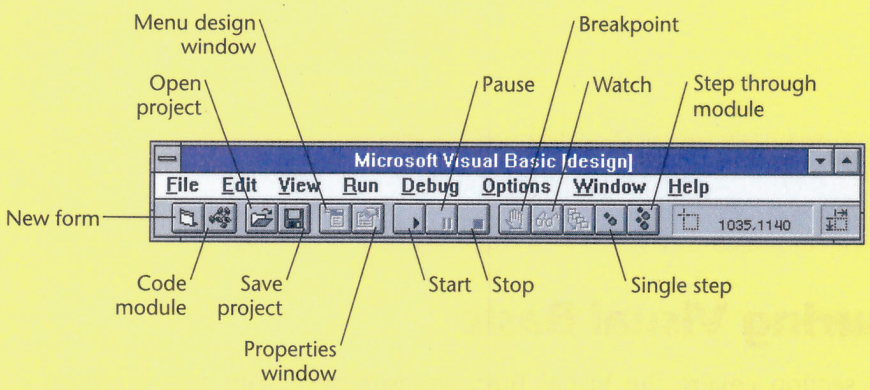


Figure 2-2
Visual Basic menu bar and toolbar

FORM WINDOW

The form dominates the center of the screen and is empty except for the caption at the top (Figure 2-3). This caption is set to the default form name: Form1. Each additional form you open is numbered sequentially. Try opening another form by clicking on the New Form icon (or selecting New Form from the File menu). Now close it.

The form is a window that contains visual objects (such as a picture box or buttons) and code. You can think of it as the "face" of your program, because it is what people see when they run the program. If you place a picture box in the upper-left corner of the form, for example, the picture will appear in the upper-left corner during run-time.

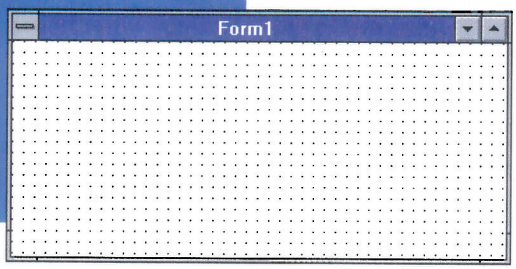


Figure 2-3
Form1

Forms serve a second purpose besides appearance. People interact with your program through the form. For example, someone could enter the name of their favorite country in a box designed for text. Or someone could close the form (and the program) by clicking on an Exit button.

PROJECT WINDOW

Forms are saved in projects. If you need only one form in your program, then that project will contain only a single form. Depending on what you are doing, though, you may need more than one form.

Why more than one? You use forms to organize how people experience your program and how they enter information. For example, imagine that you are developing a program that explains different kinds of musical instruments.

Putting all the information about different instruments on one form would probably be confusing. An alternate approach would be to use an opening form to list the instruments and ask the user to pick one. Then you could create a form for each instrument. If a person picked the guitar, for example, clicking OK on the first form would open a second form about guitars. If someone picked the piano, then the form you created for pianos would open (Figure 2-4).

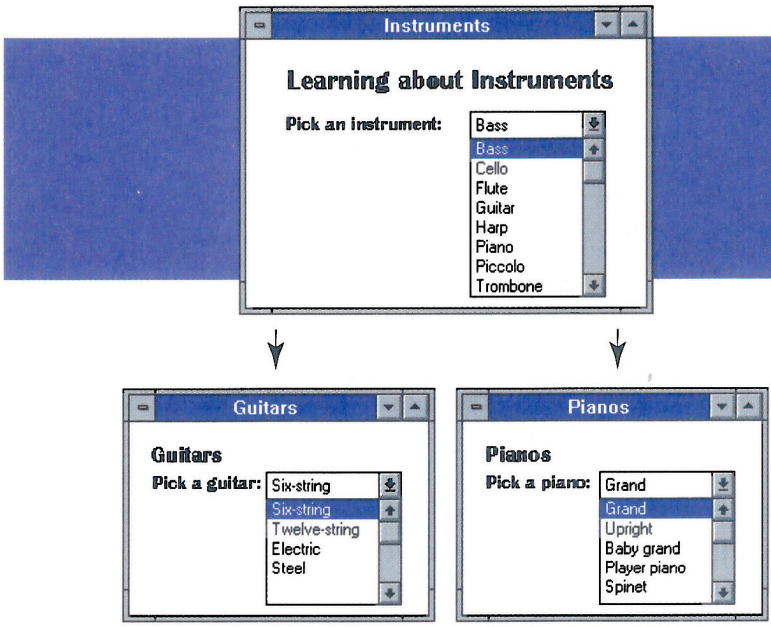


Figure 2-4
Using more than one form in a project

You may be working on more than one project at a time. Visual Basic uses the Project window to help you keep track of these projects. The window lists the project you have open and all the elements in it (Figure 2-5). Right now, you just need to be concerned with the forms in a project. Later you will learn about the code modules and custom controls that can be listed here too.

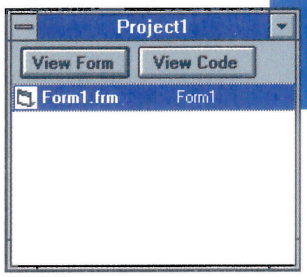


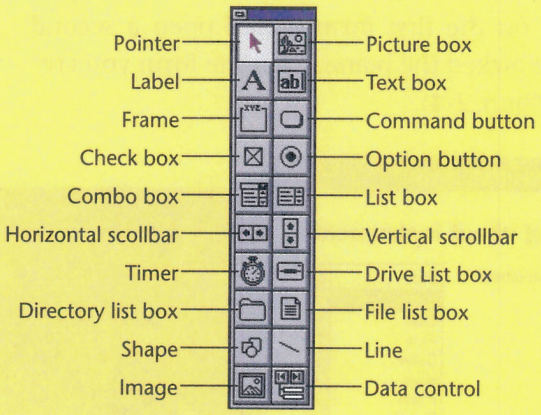
Figure 2-5
Forms in a project listed in the Project window

The first project you open is called Project1. If you do not specify another project for Visual Basic to open, or when you open a new project, Visual Basic loads a project named **autoload.mak**. This project resides in Visual Basic's own directory and contains a list of controls, the size of the Project window, and some other assumed values.

TOOLBOX

Down the left edge of the screen is the toolbox. This toolbox contains objects for building programs in Visual Basic. Look at the Toolbox now to become familiar with the different kinds of objects (Figure 2-6).

Figure 2-6
Visual Basic Toolbox



Find the picture box or image box in the toolbox. Click on it, then move the mouse cursor over Form1. The cursor now looks like two crossed lines. Click and drag to the size you want, then release the button. You have now placed an object on your form. If you imported a picture into the box, then people would see the picture when they run the form. You did not have to write any code to have this happen.

With the toolbox, you can create all the familiar elements in a Windows program. You can create icons, for example, for a toolbar. You can create a menu bar. You can create an Exit button that closes the form. You are not limited to "traditional" approaches, either. For example, you could add a button that, when clicked, covers the form in exploding blue stars.

Now try placing a second object. Find the textbox object in the toolbox. Click on it, then move the mouse cursor into the form. Click and drag to size the textbox, then release (Figure 2-7).

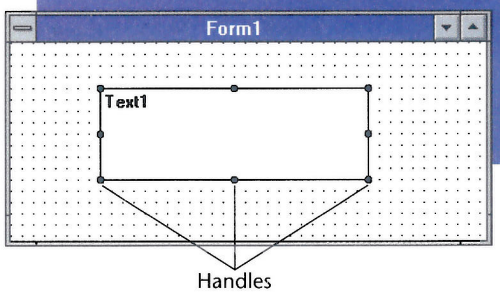


Figure 2-7
Textbox placed on Form1

PROPERTIES WINDOW

One of the characteristics of an object is that it has a "state." This state is made up of a number of properties. To understand this idea, you can think of the state of a common object, such as a glass of water on a desk. The state of that glass includes its position on the desk, its height, its width, whether there is liquid in it, and so forth.

Visual Basic lists the properties of objects in a Properties window. At any one time, the window shows the properties of only one object. Look in the Properties window to see which properties are displayed. Do the properties there belong to the form or the textbox?

The properties shown in the window belong to the selected object. The black "handle bars" around the textbox you just placed show it is selected. Click anywhere in the form to select the form, which is also an object. Now look in the Properties window. You should see **Form1 Form** at the top of the window. How have the properties changed (see Figure 2-8)?

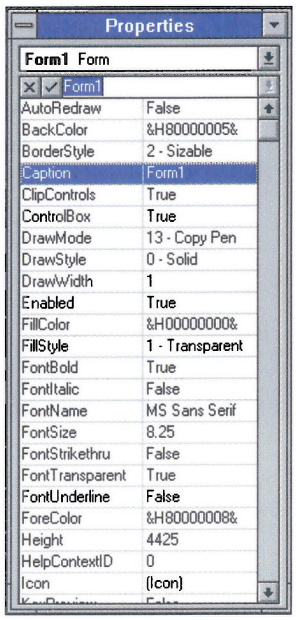


Figure 2-8
Properties of Form1

Just as you can fill an empty glass with water, you can change the properties of Visual Basic objects. To insert text into your textbox, for example, you would change the Text property. You change another property to make the text italic. Figure 2-9 shows a form with six textboxes. In each one, the same text has been inserted, but properties have been set to make the text look very different from box to box. You will experiment with changing the properties of objects in a later section of this chapter.

Figure 2-9
Six textboxes

Getting Help

Online help is available through Visual Basic's Help system. Click on Help in the menu bar to open the Help menu. From this menu, select Contents to show the table of contents for the Help system. To find information about a particular word, you would choose Search For Help On, then enter a word. Try the word "menu." Figure 2-10 shows the information on menus that is available. Double-click on the entry you want, then click on Go To.

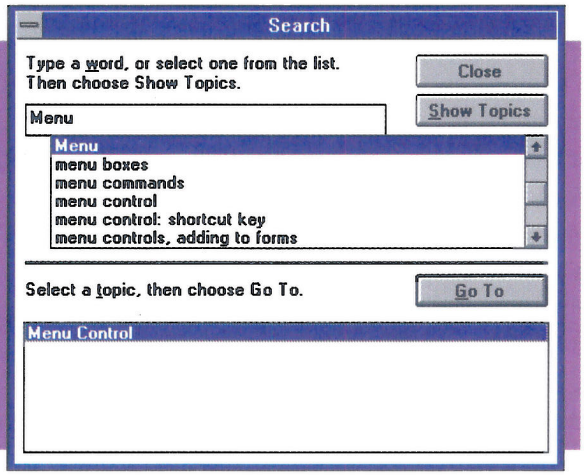


Figure 2-10
Help topics on search word "menu"

Another way to learn is to try working through a tutorial. Tutorials cover common tasks in Visual Basic, such as creating menus or debugging. Select Learning Microsoft Visual Basic from the Help menu to see your choices.

Do not forget to explore the Help system. Once you become familiar with the kinds of information available there, you can find answers to your questions quickly.

QUESTIONS

1. When you open Visual Basic, you see a number of windows. Without looking at the text or your computer, sketch the opening screen with all of the windows. Check your work by opening Visual Basic and comparing. How many did you miss?
2. Where can you find each of the following:
   - ◎ A list of an object's properties
   - ◎ A list of the forms and code modules of a project
   - ◎ A collection of icons representing objects

- A collection of buttons representing common tasks
- A collection of related objects in a project

3. For each window in the Visual Basic environment, click the arrow boxes in the upper-right corner and observe the effect. Double-click on icons to restore them to their original size. Try minimizing, maximizing, and restoring each window. Try double-clicking on the top-left box of each window to see if you can close it. Re-open any windows you close by selecting Window from the menu bar.
4. Experiment with moving the Visual Basic windows around the screen. Try resizing and rearranging them.
5. Using the Visual Basic Help system, find and read the articles on forms, events, and methods. Use the table of contents to read the article on creating the interface.

Placing Objects

In this section, you will experiment with the steps involved in placing objects. In the process, you will begin to become familiar with working in the Visual Basic environment. Every time you use Visual Basic, you will be repeating these same steps.

Starting Out

Open Visual Basic if you do not have it open. Visual Basic automatically opens a form window, as shown in Figure 2-11.

You can see the file name for the form in the Project window (Figure 2-12): **Form1.frm**. The **.frm** extension to the filename identifies the file as a form file. Does the caption at the top of the form match the default form name?

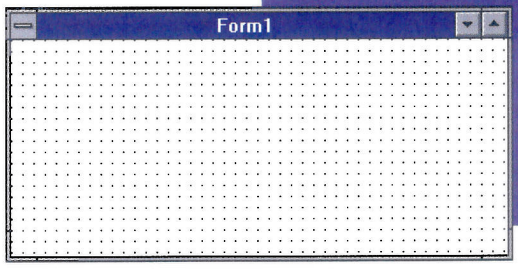


Figure 2-11
Form window

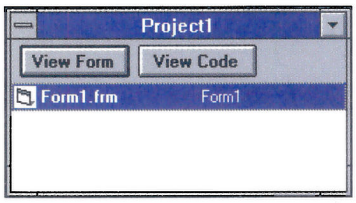


Figure 2-12
Project window

Placing the Objects

Three of the most common and useful objects for Visual Basic programs are the textbox, the label, and the command button. You will be placing these objects on the Form1 open on your screen.

Sometimes you will hear objects called controls. The two terms "object" and "control" are interchangeable. You may also hear the term "control object." All three of these names refer to the contents of the toolbox.

TEXTBOXES

Textboxes are just what they sound like: boxes that contain text. A textbox can display numbers, letters, or a mixture of both (such as "The year is 1995"). You use textboxes to accept input from people running your program. Imagine, for example, that you wanted to include a question with a yes/no answer on a form. You could put the question on the form, then place a textbox at the end of the question. A user of your program could then type yes or no in the textbox.

Create two textboxes on Form1 now (Figure 2-13). Try each of the following:

1. Double-click on the textbox icon in the toolbox. A textbox appears near the center of the form.

2. Click on the textbox icon in the toolbox. Move the mouse cursor into Form1. Click and drag to size the box, then release.

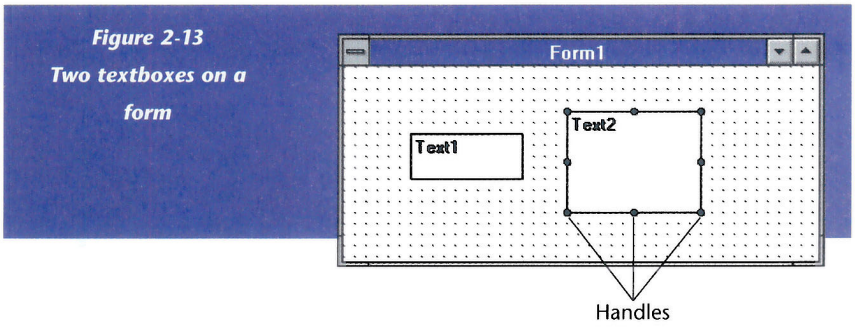


Figure 2-13
Two textboxes on a form

Now try:

3. Clicking inside one of the boxes and dragging the box to a different place on the form.
4. Dragging a corner handle to resize one of the boxes.

LABELS

You use a label object to tell something to people running your program. Users cannot change the labels you place on a form.

Try using a label as a prompting message for one of your textboxes:

1. Double-click on the label object in the toolbox. The label appears in the form. Now move the label until it is next to, but does not overlap, one of the textboxes (Figure 2-14.)

2. Be sure the label is selected (with handles around the outside edge). In the Properties window, find the Caption property for the label, then select it. What does it say?

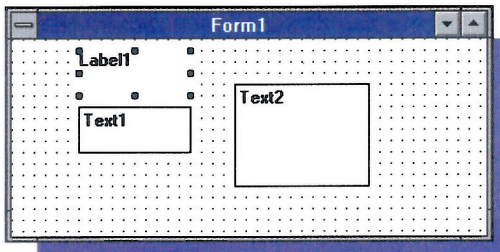


Figure 2-14
Label next to a textbox

3. Type:
   Enter a command:
   Be sure to capitalize the first "E" in "Enter." These words appear in the label box exactly as you typed them. Starting a label with a capital letter looks nicer than using lowercase.

4. See what happens to the label text when you change the size of the label box. Grab the right side of the label object and make the label smaller. Stretch it.

Now, when people run the program, they will be prompted to enter a command in the textbox. They might enter, "eat a tomato" or "put your headphones on."

A second way you can use a label is to display information from your program. For example, you could place a label on a form, then change its caption to read "You have entered the lower catacombs." This would look like the label in Figure 2-15.

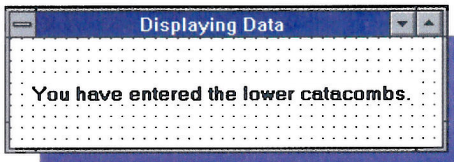


Figure 2-15
Information label

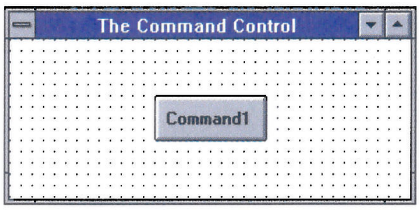
THE COMMAND CONTROL

Command controls are buttons that users click as they are running your program. When a user clicks on one of these buttons, something (an action) happens. The caption of a button, such as "Exit" or "Quit," explains what that action will be.

What makes the program close a form if the user clicks on the Exit button? How does the program know what to do? You assign code to the button, and this code is executed when a user clicks on the button.

The next section explains how to assign program code to a command button. For now, just try placing one of these buttons and changing its caption:

Figure 2-16
Placing a command button on a form



1. Double-click on the Command Control icon. A command button appears on the form (see Figure 2-16). If it overlaps with any other control, click and drag it to a new position.

2. With the command control highlighted, look at the Properties window. Be sure that the top of the window shows: Command1 Command button.
3. Find the Caption property. Change the value of this property from **Command1** to **Exit**. As you type, what happens to the command button?
4. Experiment with the size of the command button. Make it longer, then shorter, then taller. Does the caption move as the button is resized?

QUESTIONS AND ACTIVITIES

1. Use the Visual Basic Help system to search for textbox. What are the alternate names for the textbox? Why are they appropriate?
2. Change the caption of the command button to **E&xit**, then note the effect.
3. Add a new form to your project by clicking on the New Form icon on the toolbar. Remove the form by selecting its name from the Project window, then selecting Remove File from the File menu.
4. Using a magazine subscription card from a magazine as a guide, design a form to enter data for a magazine subscription. Use three controls—textbox, label, and command button—in your design.
5. Using the title page of a textbook as a guide, design a form to enter information about books. Use three controls—textbox, label, and command button—in your design.
6. What is the primary purpose of the textbox? Of the label control? Why are they often placed next to each other on a form?

7. Design a form that has the appearance of a four-function calculator. Use command buttons for the keys and a label for the display. See Figure 2-17.

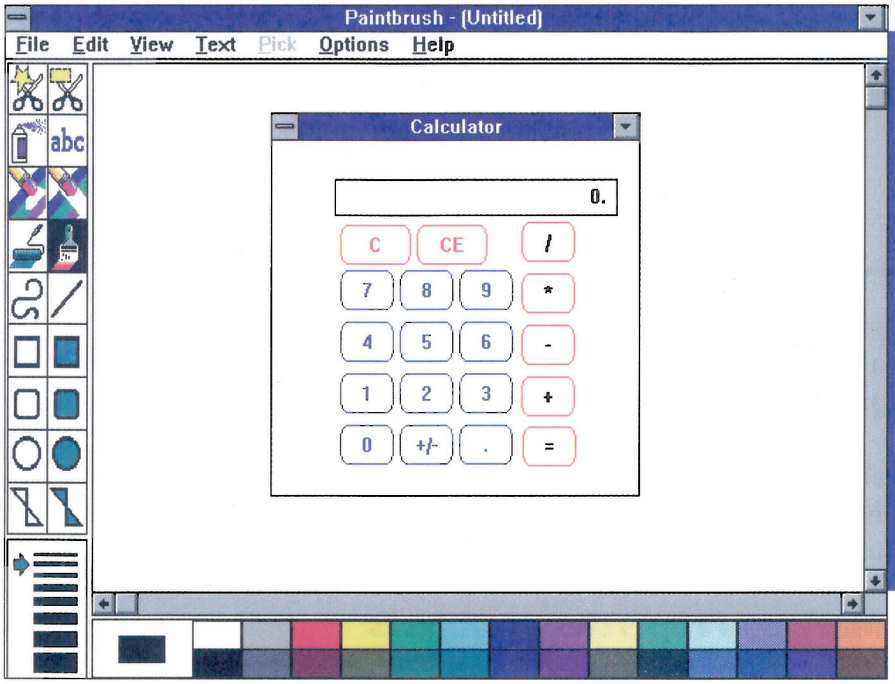


Figure 2-17
Calculator as a Visual Basic project

3
Section

Finishing a Program

After you have placed objects on a form, you typically work with their properties. These properties affect the way objects behave and appear when the program is run. For example, you may want to move three text boxes to align with each other. This could change the properties of each one. You may want to increase the size of an object, or make its caption larger and italic.

The appearance of your form is called the "look-and-feel" of the program. Part of the look-and-feel is making your program easy to understand. You want users to know what to do when they look at a form. They shouldn't have to study the form to figure it out.

One way to create an understandable form is to follow Windows conventions. Have you noticed how similar the setup is in Windows applications? Menu bars are at the top of windows, for example. The Save command is always found in the File menu. OK and Exit buttons are typically at the far-right or bottom of a dialog box. When your users

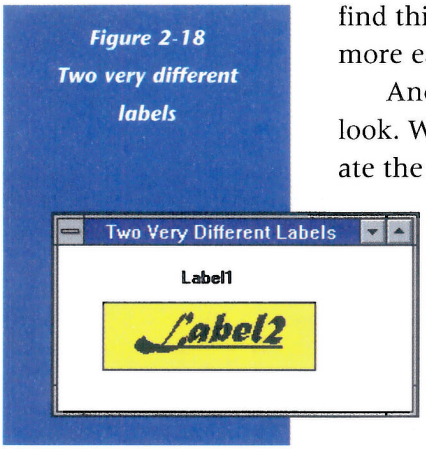


Figure 2-18
Two very different labels

find things where they expect to find them, they can use your program more easily.

Another part of the look-and-feel is how the different elements look. What is different about the two labels in Figure 2-18? Do they create the same tone, or feel?

Defining the look-and-feel of a program before writing any code is a trademark of Visual Basic. In many traditional languages, you write code first, then work with appearance. One reason Visual Basic can reverse the order is because it is a graphical environment. You can place working objects on a form without writing any code.

Generally, though, you do have some code to write or select. You want specific actions to occur when events happen, such as a user clicking on an Exit button. You need to set up the code so that the program closes the form when this event occurs.

The remaining steps in creating a program include saving, printing, and debugging the project. At the very end, you create an "executable". An executable, or **.exe** file, is what people run when they use your program.

More on Properties of Objects

In this section, you will look further at some of the properties of the three most common objects on forms: the textbox, the label, and the command button. You will become more familiar with working in the Properties window.

TEXTBOX

If you do not have a textbox on your form, place one there now. Let's look at two properties of this textbox object:

- Text
- Name

In the previous section, you had the opportunity to change the Text property of a textbox. You saw that the box automatically displays whatever value is in the Text property. The value of the Text property is the contents of the textbox.

You can change the Text property as you design a form. Usually, though, the Text property is changed by:

- A user of your program
- The program itself

For example, a user may be entering information into the program. You may have included a request on a form, such as asking users to enter their names. As the users type their names, this text becomes the new value of the Text property. While the program is running, it can read anything that users type into the Text property of textboxes. Once the names are entered, they can be saved to a file, displayed on the form, or become part of a printed page.

Your program can also use a textbox to display information on a form. As it is running, it can replace the Text property of a textbox on the form. For instance, a name entered by the user may be displayed in a textbox for further editing. In a textbox, the user can change the spelling, or add a middle initial.

An important property of any object is its name. You use the name of an object when you are writing code that uses the object. Every control, when created, has a default name. By default, textboxes are named: Text1, Text2, Text3, ...

You need to be careful not to confuse the Text property with the Name property. When you first place a textbox on a form, both the name and the Text property are the same (such as Text1). To change the contents of the box, you change the Text property. The new contents may be a few words long, such as "This is an apple."

Typically, the new name for an object is only one "word" long. See the textbox for rules on how to name objects. You should use the prefix "txt" when you change the names of textboxes (Figure 2-19):

```
txtLastName
txtBalance
txtHowManyArrows
```

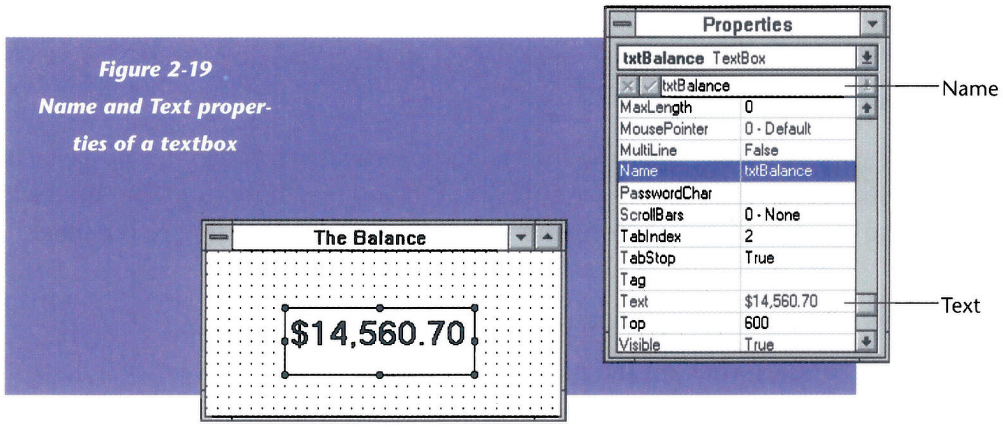


Figure 2-19
Name and Text properties of a textbox

NAMING CONVENTIONS

The way you name the controls placed on a form is important because the names are used to refer to the controls in the program code. Meaningful names make understanding the program code easier.

Textbox names start with the prefix "txt". The prefix for labels is "lbl" and for command buttons is "cmd". The prefix is followed by one or more words describing the function of the control. No spaces are used. Each new word is capitalized. Here are some examples:

```
txtLastName
txtUnpaidBalance
lblBirthplace
cmdDisplayPicture
```

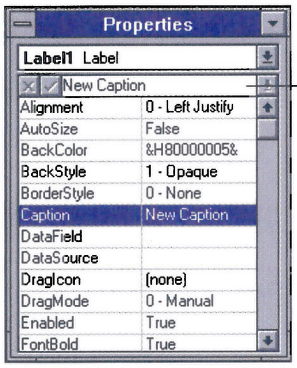
LABELS

Let's look at a few of the properties of the label object:

- Caption
- AutoSize
- BorderStyle

The caption text of a label is displayed on the screen. You experimented with changing caption text in the previous section of this chapter. Try it again now. The text automatically appears in the edit area of the Properties window (see Figure 2-20). The caption also automatically appears in the label.

Figure 2-20
Changing the caption of a label



Now change the AutoSize property of the label. When this property is set to False, the property is turned off. If you turn it on, the label fits itself to the size of the caption. The longer the caption, the longer the label.

Try both ways of changing a property value:

- Select the property and type **True**.
- Double-click on the property.

When you double-click on a property, the Properties window cycles through the different values for that property. Some properties have only two values: False and True. To change the value, then, you only

have to double-click. Other properties have more than two possible values. For these properties, you may have to double-click two or more times to find the value you want.

Look at the BorderStyle property. With the default value (0-None), no border appears around the label. Change this value so that a single-line border appears around the label on your form.

Some properties can only be altered at design time; these are listed in the Properties window. Some properties can be altered only when the program is running. Examples include SelLength, SelStart, and SelText, which are properties used to give information about selected text in a textbox. Other run-time properties are listed in the help file for each control. Most properties, though, can be changed at either time.

COMMAND BUTTON

Two important properties of a command button are:

- Caption
- Name

The caption of a command button is displayed on its face. Users click on these buttons to perform an action, such as closing a form. You want them to know what the button does, so use captions that are easy to understand. For example, use "Clear" as the caption for a button that clears the text from a textbox.

Just as with textboxes, you use the name of a command button to refer to it in code. Therefore, you need to use names that are easy to remember, such as cmdExit or cmdQuit. A button named cmdQuit should quit the program. Programmers add the "cmd" prefix to these names to indicate that the object is a command button.

Changing an Object's Properties

By this time, you have had a good deal of practice in changing the properties of objects. As a review, the steps you follow to change a property are listed here. Use these steps now to change the properties of objects on your open form.

1 Select an object by clicking on it. The object will visibly change, showing it has been selected. The Form in Figure 2-21 shows two textboxes. The second has been selected.

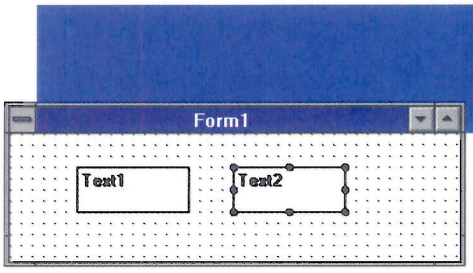


Figure 2-21
A selected textbox

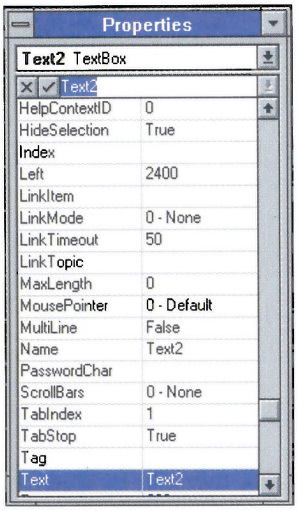


Figure 2-22
Properties window for a textbox

2 Look in the Properties window for the properties of that object (Figure 2-22). Use the scrollbar at the right of the Properties window to move up and down the list.

3 Change a property by selecting it, then typing the new value. This new value appears in the edit area at the top of the Properties window (see Figure 2-20). Or, double-click on the property to change the value.
Any change made in the property edit area will take effect immediately.

Attaching Actions to Objects

In Visual Basic, the user controls the action of the program. The user exerts control by causing events to occur. Clicking the mouse is an event. Pressing the Spacebar is an event. Controls from the Toolbox respond to events.

Different events will be introduced throughout this book. One of the most fundamental of Windows events is the Click event. The command button responds to the Click event. When the program is running and a user clicks the command button, an event procedure is executed. The event procedure is the sequence of instructions (code) run when the event occurs.

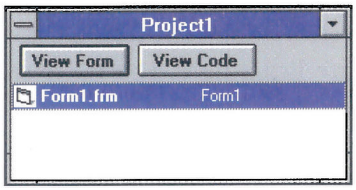
Visual Basic keeps a list of all the events to which an object responds. Let's look at that list for a command button object:

1 Double-click the Command Control icon to place a command button on the form.

2 Find the Project window. If the window is not visible on the screen, select Window from the menu bar, then select Project. The window opens on the screen.

3 In the Project window, select the name of the form.

Figure 2-23
Project window



4 Click on the View Code button (Figure 2-23).

The Code window for the form opens (see Figure 2-24).

5 In the Code window, click on the downward-pointing arrow by the Object textbox. Visual Basic opens a drop-down list containing the names of all the objects that you have placed on your form (Figure 2-24).

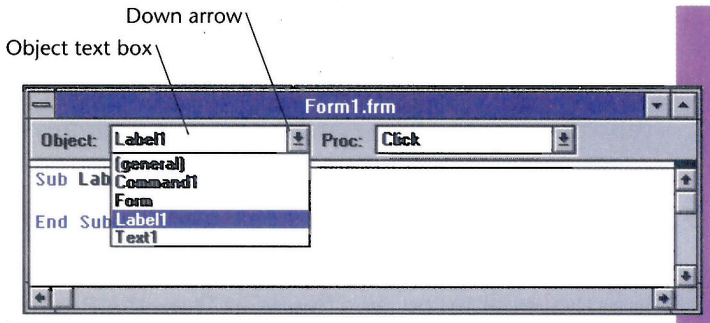


Figure 2-24
Code window for the form

6 Select Command1. This is the default name for the command button, because you have not yet changed it. The Code window displays an empty subroutine for this object. A subroutine is a section of code with one specific purpose. Another word for a subroutine is a procedure. In Visual Basic, code written for events is called an event procedure.

The subroutine is empty because it contains no instructions (statements). The name of the event to which the object responds is shown in the first line (Figure 2-25). By default, this is the Click event.

7 Look at other possible events besides the Click event. Click on the downward-pointing arrow by the Proc textbox. In the drop-down list, Visual Basic displays a list of possible events to which the command button could respond (Figure 2-26). Select one or two, and see how the first line of the subroutine changes.

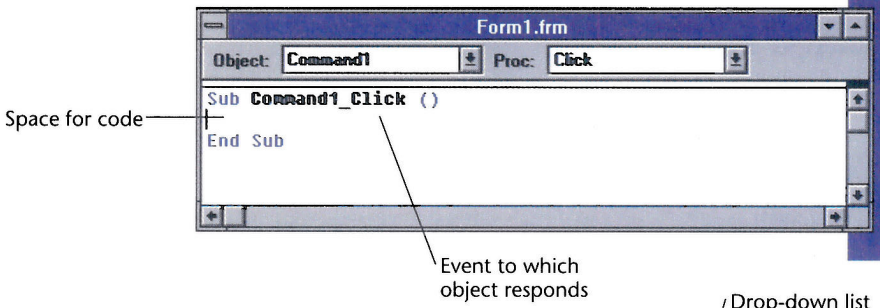


Figure 2-25
Empty subroutine for the command button

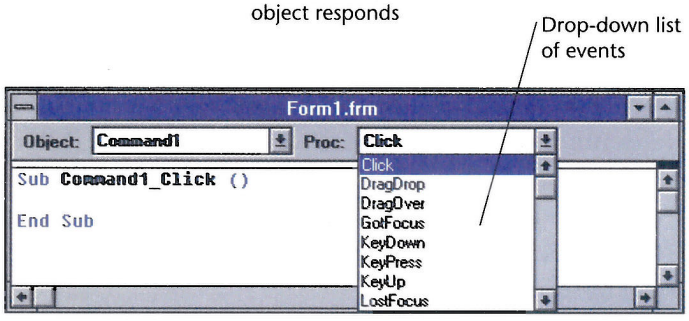


Figure 2-26
Events to which a command button can respond

8 Change the event in the Proc textbox back to Click.
The Click event occurs whenever the mouse button is clicked. Whenever the mouse is clicked on the command button, the code in this subroutine will be run. If you do not put any code in the subroutine, no action is taken by the program.

9 Between the lines of the Click Event procedure, insert the line **MsgBox "Hi there!"**

10 Run the program by selecting Start from the Run menu or by pressing the F5 key. Click the command button and note the results.

11 Stop the program by selecting End from the Run menu.

Now experiment with other objects you have placed on your form. In the Code window, open the Object drop-down list and pick a different object.

NOTE:

In Visual Basic version 2, the Change event is the default event for the label object; in versions 3 and 4, the default is the Click event.

THE KEYPRESS EVENT

One of the events to which many controls can respond is the KeyPress event. It is often used to process changes in textboxes. It is seldom used with the command button. Select this event from the Proc drop-down list in the Code window. If you look at the KeyPress event procedure, you'll see the parentheses after the name are not empty. The parentheses surround names of values used by the event procedure (Figure 2-27).

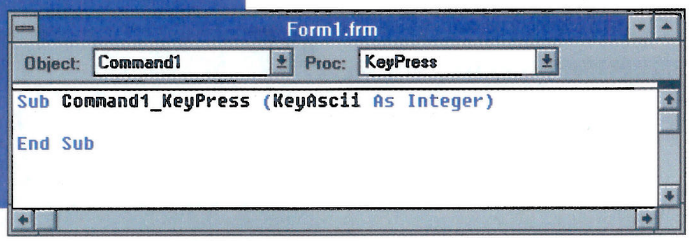


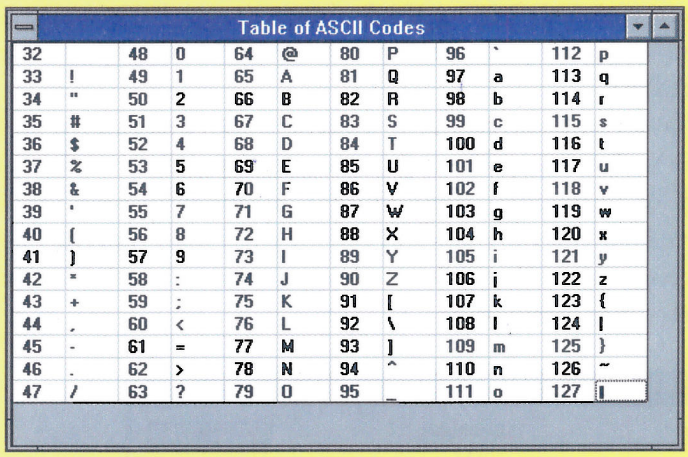
Figure 2-27 Selecting the KeyPress event for a command button

Now, when a user presses a key, the KeyPress event provides the procedure with the ASCII code for the character represented by the key. For instance, if the user presses the "A" key, the KeyPress event passes the code 65, which is the ASCII code for the uppercase A character. Once the value has been passed to the procedure, it can be used to determine what processing takes place (Figure 2-28).

Pressing certain keys on the keyboard does not cause a KeyPress event. The function keys and the arrow keys do not cause the KeyPress event to occur. Only pressing the keys used to enter text initiates the KeyPress event.

ASCII CODES

Early in the history of telecommunications, people realized they needed a standard code to represent characters and commands transmitted over wires. The American Standard Code for Information Interchange is a seven-bit code that is still in use today. These codes let people communicate and exchange information electronically, regardless of the make and model of equipment they are using.



32		48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	

Figure 2-28
ASCII codes

Creating Directories for Your Files

Before saving a project, you need to create a subdirectory to store the Visual Basic files. Create a directory with the File Manager. If you have the File Manager open, you can move from Visual Basic to the File Manager using the System menu. Open the System menu by clicking on the upper-left corner of any window. Select Switch to, then File Manager. If File Manager is closed, you should select Program Manager from the Switch to menu. Alternatively, press Alt+Tab to cycle through the open applications.

Make the new directory in **C:**, the root directory of the disk. Name the new subdirectory **VBFiles**. Return to Visual Basic via the System menu.

A similar box appears to let you save the project file. Giving the same name to the form and to the project is fine, because the two kinds of files have different extensions. In projects with several forms, each form should have a name descriptive of its use.

Saving a Project

Transferring a project from RAM to external storage is called saving. You can reload and run saved projects at any time. Select File from the menu bar. You can see that there are two commands for saving a project:

- Save Project
- Save Project As

Save Project saves the project file, the form file(s), and any other optional files comprising your Visual Basic project, with whatever names are currently active. If you have changed the name Form1 to MyForm, for example, the form file will be saved with the name **myform.frm**.

Try saving your project now (Figure 2-29). Select Save Project. The first time you save a project, you are prompted to save the changes to the form or forms first. Then you are prompted to save the files. You may have changed the names of the forms in the Properties window. If you have not, then Visual Basic assumes you want to save these forms as Form1, Form2, and so on.

NOTE:

As with any Windows program, the Save As dialog box opens if you are saving a file for the first time.

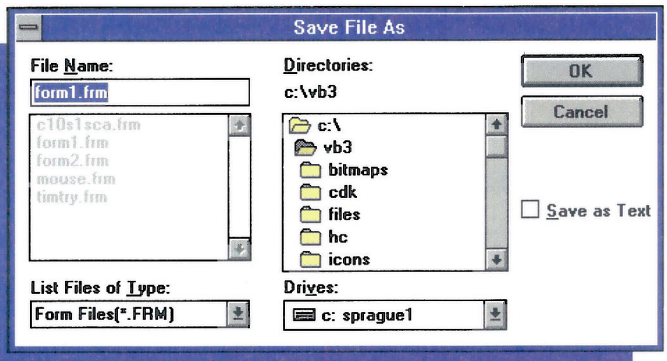


Figure 2-29
Save File As dialog box

After you have saved the forms, you are prompted to save the project itself. The default name for the first project you create is **project1.mak**. You should change this name to something more descriptive. Save all of these files in the directory you created in the File Manager to store Visual Basic programs.

In the directory window, click on **c:** to ascend one branch of the directory tree. Use the scroll bar to find your directory. Move the cursor to the File Name box and type the file name. Click OK or press Enter to save the form.

You use Save Project As if you want to create a copy of a project. Perhaps, for example, you want to experiment with a certain approach, but you don't want to lose the work you've already done. You could save your project as a different name, then experiment. You will be prompted to select different names for the project file as well as any forms or other files contained in the project.

Printing a Project

Printing your project means printing an image of the form, a listing of the code, or a listing of the properties of the controls in the project. Each component is printed on a separate page. There may be more than one form, several event procedures (which form the code), and many controls to print.

To print a component of a project, select Print from the File menu. The Print dialog box shown in Figure 2-30 opens.

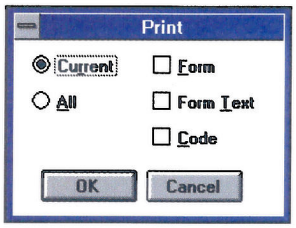


Figure 2-30
Print dialog box

You can print from either a current form or code module, or from all the forms and code modules in a project. The Form option prints a visual representation of the form. The figures shown in this chapter are similar to what you would see. Select Form Text to print text information about the objects included on the form. The result is a list showing the values of properties that have been changed from the default. Select the Code option to list the code for each event procedure.

You can select any or all of the options on the Print dialog box.

Running a Program

Running a program means to give the program control of the computer. Even a form with no controls or code may be run. When it is running other activities of the computer are suspended. The Properties window and the toolbox disappear, emphasizing the differences between run-time and design-time behavior: properties of controls may not be changed through the Properties window when the program is running; the toolbox cannot be used to add controls to a form. Properties can be changed by actions taken in the program code.

There are three ways to run a program:

- Select Run from the Visual Basic menu bar, then select Start.
- Press the function key, F5.
- Click the Start button from the Visual Basic toolbar.

Debugging a Project

Programs have flaws; some large, some small. Occasionally, these flaws, called bugs, keep the program from running. A bug may cause the program to give the wrong answer or perform the wrong action. Finding these problems can be a challenge.

Visual Basic has a number of excellent debugging tools. These tools and common debugging methods will be introduced throughout the text.

Creating an Executable File

Projects, while under development, are usually run from within the Visual Basic environment. The project is opened, loaded into memory, and run by the Visual Basic program. If you need to make additions or corrections, you can do so from within this environment.

Eventually, though, you will finish your project and move on to another one. Finished projects are ready to be run from the Program or File Manager in Windows. Preparing a program to run directly from the Windows environment is called creating an executable file.

To create an executable:

1. From the File menu in Visual Basic, open a project.
2. From the File menu, select Make EXE File.
   Visual Basic automatically converts the current project into an executable file. A dialog box prompting you for the location and name of the file appears. The default name of the file is the project name with the file extension **.exe**.
3. To run the file, double-click its name in the File Manager. Or run the file from the Program Manager by selecting Run from the File menu and entering its path and file name.

An executable program created in this way cannot be run from DOS; it must be run in the Windows environment. In addition, the appropriate Dynamic Link Library file (DLL) must be present in the System directory of Windows. For VB version 2, the filename of the DLL is **VBRUN200.DLL**. For version 3 the filename is **VBRUN300.DLL**. These files, which contain procedures necessary for the program to run, may be freely distributed with the programs you write.

QUESTIONS AND ACTIVITIES

1. What is the relationship between objects and properties?
2. What is the relationship between objects and events?

3. Explain the difference between the Name property of a textbox and its Text property.
4. Explain why the AutoSize property of the label control helps the programmer place the label accurately on the form.
5. When you buy gas for a car with a credit card, a great deal of information is recorded on the receipt. Your name, account number, the number of gallons, the price per gallon, and the total cost are all included. Design a form to display this data. Use labels for values that are display-only. Use textboxes for any information entered during the transaction (like number of gallons bought).
6. Open a form. Double-click on the textbox icon in the toolbox. Put the text "Hello, World!" in the Text property and experiment with each of the font properties (including FontName, FontSize, and FontItalic). For the same textbox, experiment with:
   - Different values of the ScrollBars property
   - Color properties
7. Using the Visual Basic Help system, look up and read definitions for the GotFocus, KeyDown, KeyUp, and LostFocus events. Write a few words describing each event.
8. Use the ASCII table in the text to encode your first name.
9. Saving the simplest project involves saving two kinds of files. Describe those files.
10. Using the Visual Basic Help system, search for help on printing code. Read the article and summarize the print choices available.

"Play Ball!"

Now that you have toured through the Visual Basic environment, it is time to create your first real project. Like the classic "Hello, World!" example, used to introduce the C programming language, this section presents a simple project. The program you will create displays "Play Ball!" on a form, along with a button to stop the program (Figure 2-31).

Figure 2-31
The baseball project

Creating the Program

The first steps in creating this project involve putting the necessary objects on the form:

1. Start Visual Basic. If Visual Basic is already running, select New Project from the File menu to create a new project. Save your old project if necessary.

2. Place a label control on the form (Figure 2-32). Double-click the label icon in the toolbox.

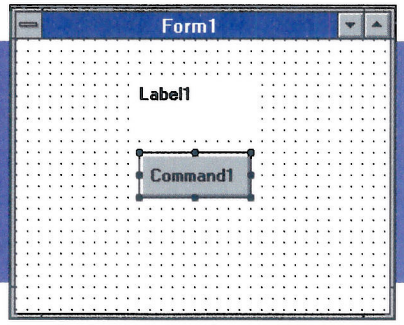


Figure 2-32
Label placed on Form1

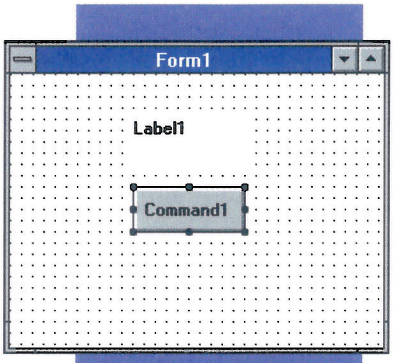


Figure 2-33
Command button placed on Form1

3. Move the label from the middle by dragging it.

4. Place a command button on the form (Figure 2-33). Double-click its icon in the toolbox.

You have now finished putting controls in place. You are ready to move on to the properties of the objects. Start with the form itself.

1. Click on the Properties window. The top line of the window shows the object currently selected. Pull down the menu to select the form (Figure 2-34).

   You can also select the form by clicking on the body of the form.

2. Select the Caption property of the form. Change the name of the form to: **The BaseBall Project** (Figure 2-35).

3. Select the label object in the Properties window.

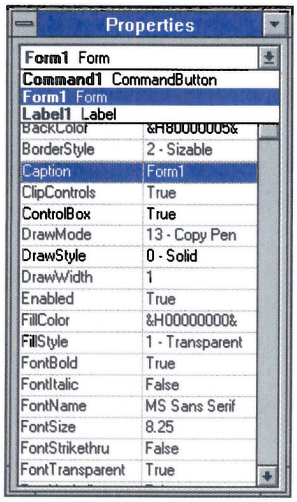


Figure 2-34
Selecting the form

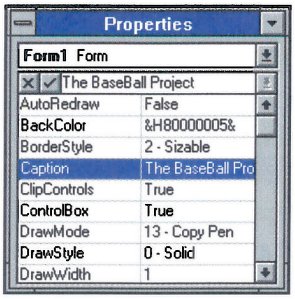


Figure 2-35
Giving the form a new name

4 Change the Caption property to: **Play Ball!** (Figure 2-36).

5 Select the FontName property of the label control. Pull down the font list from the edit area of the Properties window. The fonts listed are determined by the fonts installed in your system. Choose a font that pleases you.

6 Choose other font attributes. FontItalic, FontBold, FontUnderline, and ForeColor are all properties that affect the appearance of the display.

7 FontSize determines the size of the characters. Choose a size larger than the default for impact (Figure 2-37).

8 Once the size takes effect, the caption may no longer be wholly visible (Figure 2-38).

9 Either set the label's AutoSize property to True or resize the label control using the mouse. Once the control is resized, adjust its position on the form (Figure 2-39).

10 Select the command button in the Properties window and change its caption to **OK** (Figure 2-40).

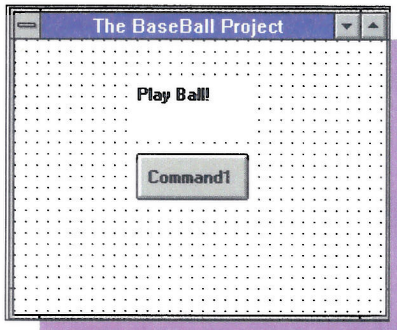


Figure 2-36
Changing a label caption

Figure 2-37
Changing caption size

Figure 2-38
Too small a label for the caption

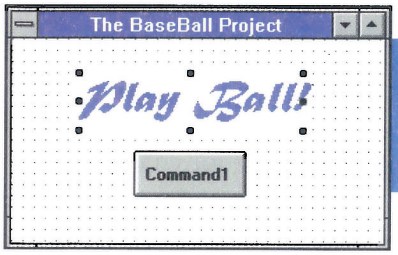


Figure 2-39
Resizing the label

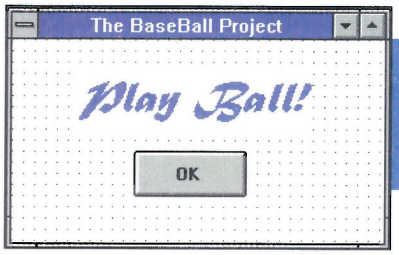


Figure 2-40
Changing a command button caption

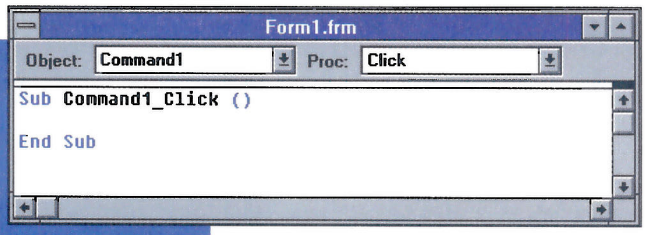


Figure 2-41
Opening the Code window

11 Double-click on the command button to open the Code window (Figure 2-41).

Between the two lines of the Click event subroutine, type **End**. When executed, this command halts the program.

That's it! Your program is ready to run, save, and print. Print the form, the code, and the form text, the listing of the controls and their properties. You covered all of these tasks in the last section. Do them now for the baseball project. If you need to, refer back to the previous section. Then, create an executable.

Stopping the Program

There are three ways to stop your program:

- Select Run from the Visual Basic menu bar, then select End from the menu.
- Click the Stop button in the toolbar.
- Click the OK button on the program's form.

Once the program is stopped, Visual Basic re-enters design mode. The Properties window and the toolbox reappear.

Loading the Project Again

A project can be reloaded for alterations or just to run it. Most programs are continually modified. Reloading the program returns the project to design mode for changes and enhancements (Figure 2-42).

Start Visual Basic. From the File menu, select Open Project.

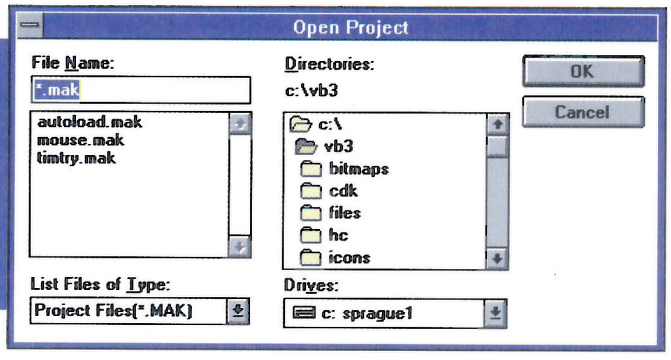


Figure 2-42
Reloading a Visual Basic project

Click your way to the directory you created and double-click the project name of your program. When the program is loaded, the form is not visible. To see the form, select the form's name in the Project window. Click on View Form in the same window to see the original form.

QUESTIONS AND ACTIVITIES

1. Change the BorderStyle property of a label.
2. With a program running, resize the main form.
3. Change a form's BorderStyle. Cycle through the choices, running the program each time and noting the effect. Which do you prefer?
4. Change the MousePointer property of a form. Once again, cycle through the choices, running the program and noting the effect. Which do you prefer?
5. Change the WindowState property of a form, cycling through the choices and running the program each time. Which do you prefer?
6. In a label control, experiment with different foreground and background colors, the ForeColor and BackColor properties.
7. In design mode (program not running), change the shape of the command button. Widen it to touch both of the form's vertical borders.
8. Cycle through the values for the Alignment property of a label. What is the effect of the different values?

Summary

The toolbar, across the top of the screen, contains buttons for commonly used Visual Basic operations.

The toolbox, on the left side of the screen, contains objects or controls that are the tools of Visual Basic. Controls are put on forms where they display and accept data and other operations.

The Project window lists the files, forms, and modules that make up a programming project.

The Properties window displays the properties of objects. You can edit these properties in the window.

The Visual Basic Help system is an excellent online resource for the VB programmer.

The form is the "face" of your program. A project may contain many forms.

The textbox control is used to enter text. The text in the box can be edited with all the customary Windows editing commands.

The label control is used to display text. Values are converted to text to display in a label.

The command control responds to a mouse click by executing the code contained in the control.

The Name property of controls allows the control to be referenced in code. The name is used to manipulate the control in Basic statements.

The Text property of a textbox can be read by a program, allowing the user to enter text. The program can also write to the Text property, turning the textbox into a display control.

The Caption property of a label control displays strings written into the property by program statements.

The Caption property of the command button is seldom changed while the program is running. The caption displays the purpose of the button.

The user causes events to which the application is programmed to respond. Each VB control responds to one or more events. Each event is associated with a section of code to handle the event. If the code is not present, the event goes unnoticed by the program.

The Click event responds when the mouse button is clicked on the control. The KeyPress event responds when a text key is pressed.

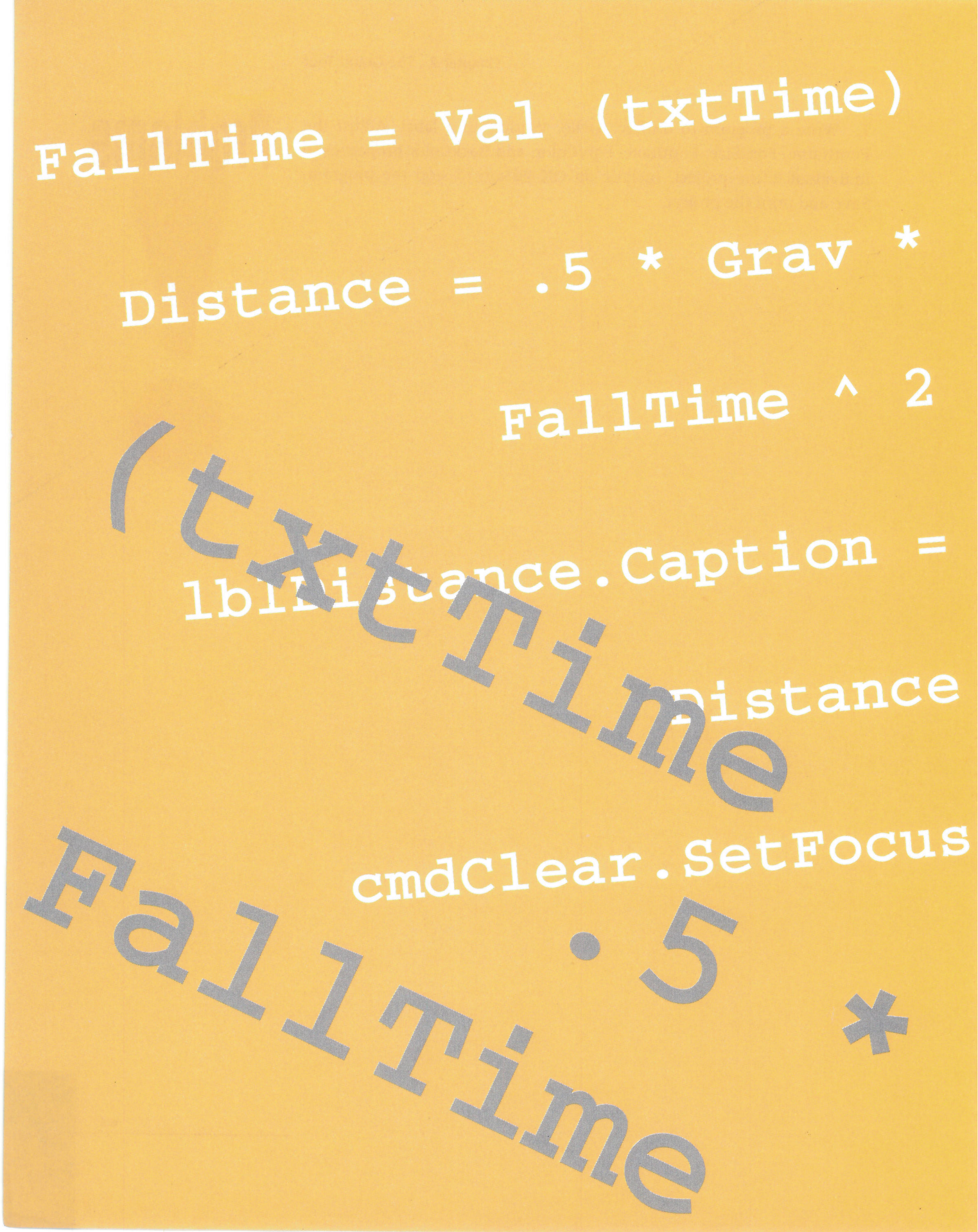
Saving a project means saving the forms, files, and modules that make up a project. Simple projects have only a single form and the project file to save.

Visual Basic can print the form of a project, just the code, or a text description of the form, called form text.

You can create an executable file from the File menu. The **.exe** file created requires the appropriate copy of **VBRUNxxx.DLL** to run.

To stop a running program, click on the Stop button in the toolbar, select End from the Visual Basic Run menu, or click on a properly coded command button in the program.

1. Write a program to display "Hello, World!" in a label. Adjust the FontName, FontSize, FontItalic, ForeColor, and BackColor properties to individualize the project. Include an OK button to stop the program. Save and print the project.


FallTime = Val (txtTime)
Distance = .5 * Grav *
FallTime ^ 2
lblDistance.Caption =
Distance
cmdClear.SetFocus
(txtTime
FallTime
.5
*