BONUS CHAPTER 3

Creating Multimedia Programs

Using multimedia techniques in your programs allows you to be very creative. You can easily create flashy programs with sound and graphic effects. Applications that use multimedia elements are becoming more and more common as the processor speeds and disk capacities of PCs continue to reach new levels. ■

What is multimedia?

Learn the different elements used in creating multimedia applications.

How to access multimedia files from Visual Basic

See how to use the `Multimedia` control to open and play a variety of multimedia files.

Using animation in your programs

See how to create simple animations that can be used to enhance your programs.

Using sound in your programs

Learn how to play sounds from within Visual Basic.

Working with Multimedia Elements

When you use the term multimedia, most people probably think of movies played on the computer with an accompanying soundtrack, or of graphics-intensive, interactive computer games. Some people might even think of the virtual reality simulations that are becoming more prevalent. While these are all uses of multimedia applications, multimedia itself has a much simpler definition. Multimedia is the use of different visual and audio techniques to present information or entertainment to the user. These techniques can include various types of audio material, animation, video, simple graphics, and even text. To give you a better understanding of multimedia as a whole, take a look at each of the elements of multimedia: sight, sound, and animation. A multimedia application combines these elements to provide an innovative and interesting user interface. Within each of the three broad categories are several elements. The discussion of these elements is the subject of this section.

Working with Sound

Sound is probably the most familiar of all the multimedia elements. One simple example of working with sound is the humble beep. You often encounter a beep when you hit the end of a data entry field or try to click a window while a modal dialog box is being shown. While you would probably realize that you could not enter any more characters or move to another window after the computer refuses to respond to your request, the beep has become a way of notifying you that you have tried to perform an operation that is not allowed. This illustrates the way that sound is used in many applications, as a way to draw attention to the computer and to the task at hand.

As you might know, you can also configure your computer to play different sounds when different tasks are performed in Windows, and in response to different Windows messages. While the use of sounds in this manner is definitely not essential, the aural cues provide notification to the user even if he or she is not looking at the monitor. In addition, if you have a CD-ROM drive, most computers allow you to play music CDs by using a CD Player application like the one shown in Figure B3.1.

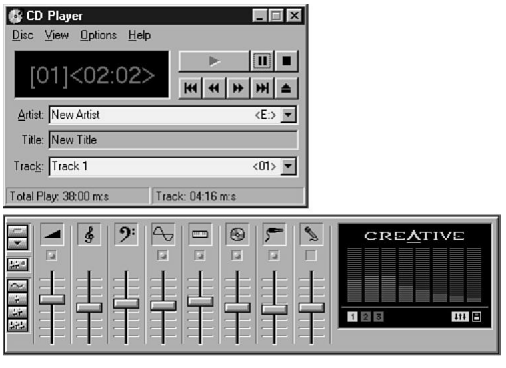


FIG. B3.1
Graphical interfaces to multimedia devices are provided by Windows 95's CD Player and a sound card mixer.

A CD Player application mimics what a real CD player looks like. When you use the software, you are actually controlling a real-world device—your CD-ROM drive. This metaphor is extended to your sound card, which has several devices used to produce various sound effects. These devices are usually manipulated in a mixer application (also shown in Figure B3.1) that comes with your sound card.

The sound that you can typically use with your computer is divided into three key types: MIDI sound effects, WaveForm sounds, and CD Audio. Let's take a closer look at these three types of sound, each of which is controllable from Visual Basic.

MIDI MIDI is the abbreviation for Musical Instrument Digital Interface. MIDI provides a way to store the notes of an instrument (or instruments) in a digital format. The MIDI file actually contains instructions for use by an internal or external *sequencer*. The sequencer interprets the instructions and synthesizes the notes that are contained in the file. Typically, MIDI files are played through the sound card of your computer and output through the speakers. However, with the proper equipment, MIDI also provides you with a way to control instruments, such as electronic keyboards, and to accept input from MIDI instruments for storage.

NOTE The sound quality of MIDI files depends on the sound card or device on which they are played. Some sound cards use a table of digitally sampled instruments, whereas others use a cruder sounding FM synthesizer. Keep this in mind when distributing a MIDI file. ■

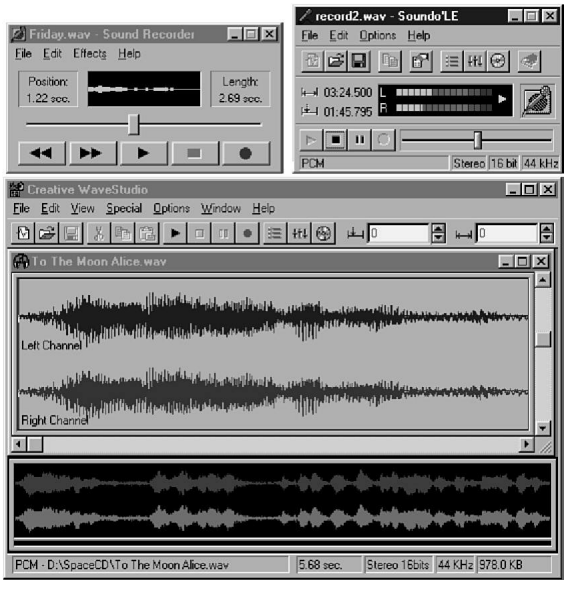
The audio files for MIDI sounds are much smaller than similar files for WaveForm audio. This is because the MIDI file only contains instructions on how to create the note as opposed to actual sound samples. (This is similar to the differences between bitmap and raster graphics.) MIDI files are great for providing musical backgrounds for an application or Web page. The disadvantage of MIDI files is that you cannot use them for storing voice information, such as narrations or voice annotations, because they only simulate musical instruments.

WaveForm Audio WaveForm audio is digitized sound that has been stored in files on your computer. Because WaveForm audio is a sample of the actual sounds, it can contain music, voice, or a variety of other sound effects. WaveForm audio is one of the most common forms of audio used in a computer and one of the easiest to create. While MIDI files typically require specialized instruments to create the music, you can create WaveForm audio with a sound card and a microphone. Most sound cards that you get for your computer have the capability of accepting audio input for storage in WaveForm files. This input can come from a microphone, tape deck, or other audio source.

To record WaveForm audio, you can use the Sound Recorder program that comes with Windows 95 (shown in Figure B3.2), or a sound recorder that comes with your sound card. These programs work in the same manner as a tape recorder or VCR. To record sounds, just press the record button and start sampling the sounds you want to record. When you are finished recording, press the stop button and specify the file name and where you want to store the sounds. WaveForm audio provides an easy way to add narration to a program. You can also

store tracks from an audio CD in Wave format before writing them to CD-R (recordable) media. WaveForm files, which usually have the extension WAV, are generally much larger than MIDI files. They can be recorded at a number of different sampling rates, which range from AM-radio to CD-sound quality.

FIG. B3.2
The Windows 95 Sound Recorder and other typical sound recorder programs let you record audio for use in your programs.



CD Audio CD audio is the type of audio that you are used to hearing from your home or car stereo. The CD audio capabilities of your computer enable you to play music CDs on your computer. If you do not have a sound card, most CD drives have a headphone jack that lets you listen to the music while the CD Player program controls functions such as Play, Pause, and Skip. If you have a sound card, you can connect the CD to the sound card and play the music through the same speakers that are used for WaveForm files.

CD audio gives you the highest sound quality of any of the audio types. However, it is different from a WAV file in that it is stored on an actual physical CD. You can use VB to play CD tracks in your program. This might be useful in creating a multimedia-enhanced CD-ROM, where you could store both the program and audio tracks on the same CD-ROM. In the past few years, CD-R drives and media have fallen in price, so creating your own CD-ROMs is well within reach. Typically, 176 kilobytes of storage are required for each second of CD audio. This allows about 73 minutes of audio (or 650 megabytes of data) to be stored on a single CD-ROM.

Determining What Your System Will Support For your computer to work with any of the types of sounds listed in the previous sections, you need to verify that your computer supports the sound type. The easiest way to determine what sounds your system supports is to use the

Windows Media Player application (choose Start, Programs, Accessories, Multimedia, Media Player). After starting the Media Player, click the Device menu. This shows you what types of audio and video devices can be used on your computer (see Figure B3.3).

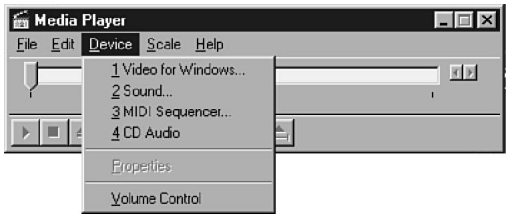


FIG. B3.3
The Windows 95 Media player (MPLAYER.EXE) lists the multimedia devices installed on your system.

The three sound types discussed earlier in this section are represented by the following menu items in the Device menu:

- *Sound* WaveForm audio
- *MIDI Sequencer* MIDI audio
- *CD Audio* CD music

Working with Graphic Elements

As there were several different types of sound that you could use in your applications, there are also several types of visual elements that can be created or used in your Visual Basic programs. These elements are digital and analog video, animation, and graphics, which includes still pictures, bitmaps, and charts. Which of these elements you use and how you use them depends on the nature of your application. Each element has different capabilities and uses in an application. Let's take a closer look at the major visual elements.

Video There are two types of video that can be used in your programs: video stored in files and video stored outside your PC. Video files are movies that have been captured by using a video capture board or digital camera and stored in a file on your computer. For motion video, a specified number of frames of information are stored per second of video. Typically, you have from 15 frames per second (fps) to 30 fps, which is considered full motion video. Digital video is the type that you find in most encyclopedia programs or tutorial disks that include video tracks. Some typical video file extensions are AVI, MPEG, and MOV.

Video can also be received from an external device such as a laser disc, camera, or video cassette recorder (VCR). In this case, the computer program might be controlling the device or displaying it on the computer screen. Because of the need for specific external devices, this type of video is more difficult to use and distribute in your programs.

Animation Animation is simply displaying a series of images to give the user the impression of motion. You have probably seen animation used in Internet Explorer, where the globe spins while you are waiting for a page to load. You might have also seen animation in the Windows Explorer when you are copying a group of files. In this case, letters flying from one folder to another indicate that an operation is occurring (see Figure B3.4).

FIG. B3.4
File operations use animation to indicate an ongoing process.

There are two basic types of animation: frame-based animation and object-based animation. Frame-based animation uses a series of frames (usually bitmaps) displayed in rapid succession to give the illusion of motion. Each frame of the animation is a unique picture that displays the entire scene of the animation. Motion is achieved by making subtle changes in the position of objects in each frame of the animation. If you have ever created a flip-book cartoon, you have used frame-based animation. Many animated cartoons use frame-based animation.

In object-based animation, each object of a scene is independent from the others. In object-based animation, an authoring program is used to change the relative placement of each object from scene to scene. This technique is also known as sprite animation. An example would be using a timer control to increment the `Left` property of a PictureBox, thus making it "move" across the screen.

Graphics Graphics are static images, in contrast to video or animation. However, graphics comprise a large portion of many multimedia applications. Graphics can include still pictures, bitmap illustrations, business charts, and even dynamic charts that change over time. (For more information about using graphics in your programs, see "Doing Graphics" on the CD-ROM.)

Working with Text

The final piece of the multimedia puzzle is text. Although text is not as glamorous as video or sound, it is a very important part of almost any program you create. Text still provides you with the most efficient way to present a large amount of information. Text also takes up less space than other media and is easier to use as reference material. Even though text is not fancy, this does not mean that it has to be boring. You can use fonts and color to liven up your text and highlight key information. You can also use specialized text files, like HTML, along with a browser to implement jumps between one document and another. This makes your text interactive to the user. Chapter 14, "Working with Text, Fonts, and Colors," covers this in more detail.

Exploring the Uses of Multimedia

As you have seen, there are a number of elements that can be involved in a multimedia program. Likewise, there are a number of uses for multimedia applications. Some of these uses include:

- *Games* Many games include multimedia elements to add to the gaming experience.
- *Entertainment* With multimedia applications, you can enjoy music, cartoons, and movies right on your computer screen.

- *Training* With the use of sound to accompany slide shows or videos, you can create excellent computer-based training programs for a number of training needs. In fact, if you look in your Windows/Media folder, you will find a number of sound-enhanced videos that demonstrate how to perform tasks in Windows.
- *Enhancing business presentations* Using sound and animation can make dry numbers come alive for your audience.
- *Information Centers* You can use multimedia elements to build kiosk applications that can direct visitors and display information.

As you begin programming multimedia applications, you will probably find more uses for them than you could have imagined.

Using Animation in Your Programs

Animation is one of the easiest multimedia effects to add to your applications. As you read earlier, animation is simply presenting a series of images in succession. You can use animation to simulate motion of an object, to indicate time passing, or to add transition effect for elements of a presentation. Many parts of the Windows operating system use animation to let the user know that a program is performing a task, such as copying a file or loading a program. Animation is also used in most screen savers to provide a little entertainment. (Some screen savers take this to the extreme by providing a series of animated cartoons. Some of these are so much fun to watch that you forget about doing other work.)

Within Visual Basic, there are two basic ways to handle animation. First, you can write code to present the image series yourself. This method provides you with the most flexibility because you control the images being shown, the timing of the images, and all other aspects of the animation. The second method for adding animation to your program is to use the Animation control.

NOTE You can also provide some very simple animation for toolbars and command buttons by changing pictures on the buttons as they are pressed. One example is to open a door on the exit button as it is clicked by the user. ■

Creating Your Own Animation Effects

To create your own animations, you need three things:

- A control to display the images
- A source for the series of images to be displayed
- A timer routine to determine when to display the next image in the series

To illustrate the concepts involved in creating animation, let's walk through the process of creating a simple animation program that creates a rotating moon on the form.

The source code and bitmaps for this project are also located on the companion CD-ROM.

Setting Up the Basic Form The first step to creating an animation sequence is to set up the basic form for displaying the images of the sequence. The form needs a control to show the images, command buttons to start and stop the animation sequence, and a text box to allow the user to enter the speed of the animation. To create the basic form, follow these steps:

1. Start a new Standard EXE project in Visual Basic.
2. Change the name of the form to `frmAnimate`.

3. Add an Image control to the form and change its name to `imgAnimate`.

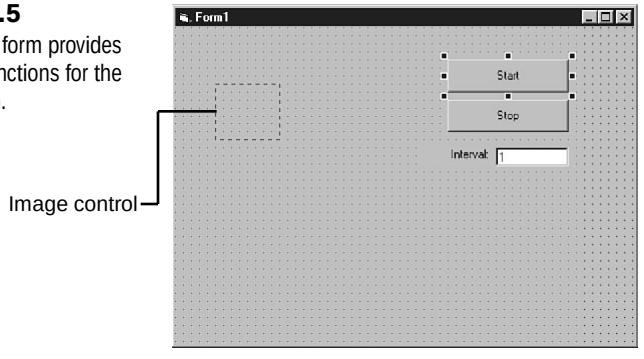
4. Add two command buttons to the form and change their names to `cmdStart` and `cmdStop`. Change the `Caption` properties of the command buttons to `Start` and `Stop`, respectively.

5. Add a text box to the form and change its name to `txtSpeed`. Set the `Text` property to `1` to set the default speed of the animation.

6. Add a `Label` control to the form with the caption `Interval:` and place the label next to the text box.

When you have finished these six steps, your form should look like the one in Figure B3.5.

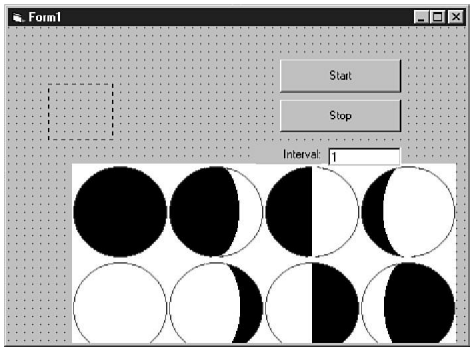
FIG. B3.5
The basic form provides control functions for the animation.



Setting Up the Image Source The next step in creating the animation program is to set up the source of the images for the series. For the demonstration, you use an array of image controls. This array is named imgSource and consists of eight image controls with the `Index` properties ranging from `0` to `7`. After you have created the control array, you need to set the picture property of each of the controls. In our example, there are eight bitmaps that display a moon in various phases. These images can be created in the Windows Paint program fairly easily.

After setting the `Picture` property of the Image controls to a different bitmap, you need to set the `Visible` property of the controls of the array to `False`. This keeps the source images from being displayed when the program is run. When you have finished setting up the source images, your form will look like the one in Figure B3.6.

FIG. B3.6
A control array holds the source images.



NOTE You could also use an `ImageList` control to store the images for the animation sequence. ■

Handling the Changing of Images The final step to creating the animation program is to set up the code to show the images in sequence. This involves setting the initial picture of the sequence and displaying subsequent pictures at a specified time interval. The first thing you need to do in the code is to set up module level variables to determine the interval for changing images, the current image sequence number, and a flag to determine whether the animation is running or stopped. These variables are declared in the Declarations section of the frmAnimate form. The declarations are shown in Listing B3.1.

Listing B3.1 FRMANIMATE.FRM—Declaring and Initializing Variables to Control the Animation Sequence

```
Dim PicNum As Integer, RunAnimate As Boolean
Dim RotInterval As Single

Private Sub Form_Load()
  imgAnimate.Picture = imgSource(0).Picture
  PicNum = 0
End Sub
```

After the variables are declared, the `Load` event of the form initializes the variables and places the first image of the sequence in the image control used for display. This is also shown in Listing B3.1.

Next, you need to write the code for the two command buttons that start and stop the animation. In the `Click` event for the cmdStart button, you retrieve the time interval between image changes from the text box and then set the run flag to indicate that the animation is running. Finally, you call the routine that actually runs the animation. The cmdStop button performs a single function: it sets the `RunAnimate` flag to `False` to terminate the animation run. The code for these two command buttons is shown in Listing B3.2.

Listing B3.2 FRMANIMATE.FRM—Command Button Code

```
Private Sub Command1_Click()
  RotInterval = Val(txtSpeed.Text)
  RunAnimate = True
  RotateMoon
End Sub

Private Sub Command2_Click()
  RunAnimate = False
End Sub
```

The final routine is the one that changes the image displayed to the next one in the sequence. The code for this routine is shown in Listing B3.3.

Listing B3.3 FRMANIMATE.FRM—Changing the Image in the Display Control to the Next One in the Animation Sequence

```
Private Sub RotateMoon()
Dim StTime As Single, CurTime As Single
Do While RunAnimate
    imgAnimate.Picture = imgSource(PicNum).Picture
    DoEvents
    StTime = Timer
    Do
        CurTime = Timer
    Loop Until CurTime > StTime + RotInterval
    PicNum = PicNum + 1
    If PicNum > 7 Then PicNum = 0
Loop
End Sub
```

As you can see in Listing B3.3, the code runs as long as the variable `RunAnimate` is set to `True`. With each pass through the loop, the `Picture` property of the `imgAnimate` control is set to the `Picture` property of one of the source image controls. The index of the control is determined by the `PicNum` variable. This variable gets incremented each time you pass through the loop. Because the `Index` property of the control array only ranges from `0` to `7`, the `PicNum` variable gets reset to `0` each time it goes past `7`. This causes the sequence to start over and avoids any errors associated with an index out of range. The interior loop is the one that delays the execution of the next pass of the loop until the specified interval has passed. `Timer` is a VB function that returns the number of seconds that have elapsed since midnight. The returned value is a

Single data type and can handle fractional seconds. The inner loop repeatedly calls the Timer function until the interval has elapsed.

One final line of note is the DoEvents statement. This statement serves two purposes. First, it gives your program a chance to update the image so the new picture is displayed. Without this, you would see a single image displayed on the form. The second purpose is to allow the user to click the Stop button to terminate the animation.

NOTE Instead of using the Timer function, you could place the code to change the images in the Timer event of a Timer control. You would then set the Interval property of the timer control to the value specified by the text box. ■

Running the Program After saving your program, press F5 or click the Start button to run the program. Set a value for the image interval and click the Start button. If you leave the interval set at 1 second, you see the images change in a discrete manner. If you set the interval to 0.1 seconds, the image changes quickly enough that the rotation of the moon appears to be continuous. Try several settings of the interval and see how it affects the behavior of the animation.

NOTE Because of executing the DoEvents statement, continuing to reduce the interval beyond a certain point produces no appreciable speed increase. For the test machine I was using, this interval was 0.05 seconds. If you require extremely fast graphics, you might want to look into using a graphics API, such as DirectX. ■

Using the Animation Control

The other method of displaying animation sequences in your programs is to use the Animation control. This control displays silent digital video files (.AVI). The control allows you to start and stop the animation process as well as determine whether the animation sequence stops at the end of the file or loops back to repeat the sequence. AVI files can be created by using a video capture card with a video camera or VCR.

▶ **See** "Using Video in Your Programs," in Chapter 11.

Exploring the Multimedia Control

In the previous sections, you learned about what multimedia is and what types of elements can be used in a multimedia application. You also learned a little bit about creating your own animation effects with code. Although the animation is important, the bulk of your multimedia work will involve accessing different types of multimedia files and playing them through the various multimedia devices on your system. For this, you need a method to open the files, access the appropriate device, and then play the files. In most cases, you also need a method to control the playback of information. For example, most people would not be too happy with a CD player that played all tracks straight through. Your users will want the ability to skip tracks, replay certain tracks, and pause the playback. All these things can be handled with your multimedia programs.

In Windows, multimedia devices are handled through the Media Control Interface (MCI). MCI is a set of Windows functions that can be used to control all aspects of working with multimedia files, from setting up the devices, to pausing playback, to providing information about the current position within the playback (for example, elapsed time). These functions can be accessed directly by declaring the appropriate functions in your Visual Basic program and making the proper function calls to handle your needs. Using the API functions provides you with the most flexibility in programming multimedia applications.

NOTE The API functions used to control multimedia devices are not covered extensively in this book. However, there are some examples of this in Chapter 35, "Accessing the External Functions: The Windows API." ■

▶ **See** "Calling Basic API and DLLs," in Chapter 46.

Fortunately, Microsoft has also provided you with an easier way to handle controlling multimedia devices. The Multimedia control (MCI32.OCX) provides you with easy access to most of the functions of the API. The control does this through properties that can be set at design time and changed at runtime. The Multimedia control also uses events to notify you of things taking place in your program, such as the end of playback. Using the Multimedia control will handle a large portion of your multimedia programming needs.

Going Through the Basics

The first step to working with the Multimedia control is to add it to your toolbox. You do this by choosing the Components item from the Project menu, checking the box next to the Microsoft Multimedia Control 5.0 item, and then clicking OK. The next step is to add the control to your form. You do this by clicking the control in the Toolbox and drawing it on the form just like any other control. Figure B3.7 shows a form with two Multimedia controls.

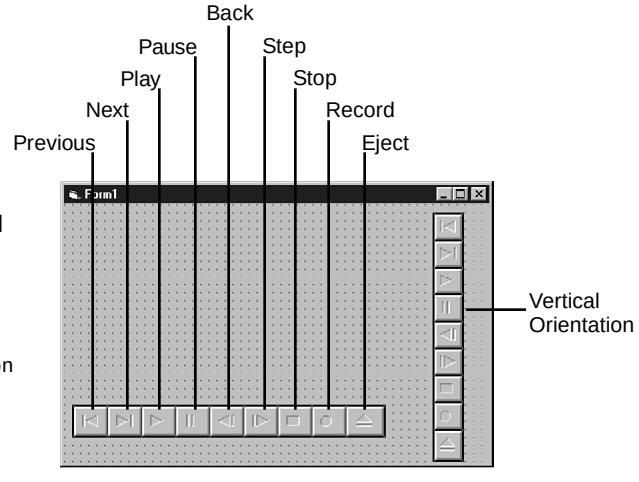


FIG. B3.7
The Multimedia control first appears in a horizontal orientation, but you can change it to vertical with the control's `Orientation` property.

There are two properties that control the basic appearance of the control. The Orientation property determines whether the control is drawn horizontally or vertically. The BorderStyle property determines whether a single line border is drawn around the buttons of the control.

As you can see in Figure B3.7, all the buttons of the control are displayed when you draw the control and all the buttons are shown as disabled. When you run a program with the Multimedia control, it automatically enables the buttons that are applicable for the type of media you are accessing. As you examine using the Multimedia control to access different types of multimedia elements, you will see which buttons are enabled for each element. For now, let's look at what each button does:

- *Previous* For devices that use tracks, such as CD Audio, moves to the beginning of the current track. If clicked a second time within three seconds, goes to the beginning of the previous track.
- *Next* For devices that use tracks, such as CD Audio, moves to the beginning of the next track.
- *Play* Begins playing the currently open device and file.
- *Pause* Pauses the playing or recording of the current device. When playing or recording is resumed (by clicking the appropriate button), resumption occurs at the point where the program was paused, not at the beginning.
- *Back* Steps backward through the file or track being played. This is the equivalent of a rewind button on a tape recorder.
- *Step* Steps forward through the file or track being played. This is the equivalent of a fast-forward button on a tape recorder.
- *Stop* Halts execution of the current playback or recording and returns the file or track pointer to the starting position.
- *Record* Begins recording sounds from an appropriate input device.
- *Eject* Issues the eject command to remove a CD from the drive.

NOTE Buttons on the control are handled automatically only if the AutoEnable property is set to True (its default setting). If you set the AutoEnable property to False, your program has to handle enabling and disabling the appropriate buttons in the control.

By default, all of the buttons are shown while the Multimedia control is in use and only the appropriate buttons are enabled. If you want to manually enable and disable buttons, or hide the buttons that are not in use, you can do so through the Enabled and Visible properties of each button. These properties are set to either True or False and work just like the Enabled and Visible properties of a regular control. Table B3.1 lists the property names for the different buttons.

Table B3.1 Using Properties to Configure the Multimedia Control Buttons

Button Name	*Enable* Property	*Visible* Property
Previous	`PrevEnabled`	`PrevVisible`
Next	`NextEnabled`	`NextVisible`
Play	`PlayEnabled`	`PlayVisible`
Pause	`PauseEnabled`	`PauseVisible`
Back	`BackEnabled`	`BackVisible`
Step	`StepEnabled`	`StepVisible`
Stop	`StopEnabled`	`StopVisible`
Record	`RecordEnabled`	`RecordVisible`
Eject	`EjectEnabled`	`EjectVisible`

Setting Up the Control

After you have drawn the Multimedia control on your form, you are ready to start working with multimedia devices. To work with any multimedia device, you need to set the `DeviceType` property of the control. You can set this property at design time or at runtime. The `DeviceType` property tells the control the type of information that it will be processing, such as a WAV file or CD Audio. Then, depending on the type of device that you are using, you might also need to specify the `FileName` property. For the devices that use files, the `FileName` property specifies the file to use for playback or recording. As with the `DeviceType` property, you can set the `FileName` property either at design time or runtime. Typically, your applications will include the capability to set the `FileName` at runtime so that multiple files can be played. Table B3.2 provides a list of the `DeviceType` settings for the Multimedia control and tells you whether a `FileName` is required for the device.

Table B3.2 Devices Supported by the Multimedia Control

Device	*DeviceType* Setting	*FileName* Required
AVI videos	`AVIVideo`	Yes
Music CDs	`CDAudio`	No
Digital Audio Tape	`DAT`	No
MIDI	`Sequencer`	Yes
Wave sounds	`WaveAudio`	Yes

Working with the Devices in Code

Setting the DeviceType and FileName properties are the only parts of the Multimedia control setup that can be done in the design environment. To complete the setup of the control, you must open the device and file that you want to use. This function, as well as control of the recording or playback, can be accomplished only by using code while your program is running. You control the multimedia devices through the use of the Command property of the Multimedia control. To open a device for use, you need to set the Command property, as shown in the following line of code:

```
mmcAudio.Command = "Open"
```

After this command is issued, the device is opened and the appropriate buttons of the Multimedia control are enabled (assuming you left the AutoEnable property set to True). If you want to use just the Multimedia control for playing sounds and movies, the Open command is the only one that you need to use. The control itself handles running the other commands necessary to play or otherwise control the multimedia device. If you want to use your own buttons or other code to control the multimedia device, you need to use other settings of the Command property. Table B3.3 lists the settings of the Command property and the corresponding button of the Multimedia control.

Table B3.3 Command Property Settings

Command	Button
Open	N/A
Close	N/A
Play	Play
Pause	Pause
Stop	Stop
Back	Back
Step	Step
Prev	Prev
Next	Next
Seek	N/A
Record	Record
Eject	Eject
Sound	N/A
Save	N/A

You might notice that several of the commands do not have corresponding buttons. These commands have special purposes, such as saving a recording to a file or closing the multimedia device. You see how some of these commands are used in the section "Creating Programs to Play Multimedia Files."

Handling the Control's Events

In addition to the property settings of the Multimedia control, it has several events that you will utilize in creating your programs. These events are in addition to the standard events that are applicable to most controls. Let's start the discussion with the button events.

Button Events For each button of the Multimedia control, there are four events: `Click`, `Completed`, `GotFocus`, and `LostFocus`. These events are identified by the button name and the event name—for example, PlayClick for the `Click` event of the Play button. The `GotFocus` and `LostFocus` events for each button work just like the similar events for any other control. Therefore, focus on the `Click` and `Completed` events.

The `Click` event of each button automatically invokes the command associated with the button. For example, when you click the Play button, play of the device automatically starts; you do not have to write code to make it happen. Therefore, you use the `Click` event to handle any other tasks that are required when the button is pressed. One other difference between the `Click` event of the buttons and that of any other control is the `Cancel` parameter that is passed to the event. By setting this parameter to `True`, you prevent the button from performing its default command.

NOTE The default command of a button is performed after all the other code in the `Click` event. ■

The `Completed` event of a button occurs when the MCI command issued by the button has finished. This allows you to write code to perform a task at the end of a command. The `Completed` event also passes an `Errorcode` parameter. This parameter tells you whether the command completed successfully. If so, the `Errorcode` parameter is `0`; otherwise, the `Errorcode` is set to a value that indicates the problem that occurred.

Other Events In addition to the button events, there are two other key events for the Multimedia control: the `Done` event and the `StatusUpdate` event. The `Done` event does for the control as a whole what the `Completed` event does for a button. The `Done` event is fired when a command has completed. The `Done` event passes the `NotifyCode` parameter to tell you the completion status of the command. The values of the `NotifyCode` parameter are shown in Table B3.4.

Table B3.4 Notification Codes for the *Done* Event

Value	Description
1	The command completed successfully.
2	The command was stopped and superseded by another command.

Value	Description
4	The command was aborted by the user.
8	The command failed.

The StatusUpdate event occurs automatically at specified intervals. This event updates the properties of the Multimedia control that will inform you of the status of the multimedia device. For example, when playing a CD, the Multimedia control keeps track of which CD track is being played and how much of the CD has been played. This information is updated when the StatusUpdate event is fired. You can use the event to display this information in your program. The interval used to fire the StatusUpdate is controlled by the UpdateInterval property. This property sets the number of milliseconds between events. The default value is 1000 milliseconds, which causes the StatusUpdate event to be fired once every second.

Creating Programs to Play Multimedia Files

The best way to demonstrate the capabilities of the Multimedia control is to write some sample programs that use the control and its commands. This section shows you the basics of creating applications to play CD Audio, AVI videos, MIDI files, and WaveForm sounds. In each section, you see how to set up the control and how to get status information from the control. You also see which buttons of the control are enabled for each type of multimedia element.

Playing CD Audio

The first multimedia element you will work with is CD Audio. Because the CD Audio does not require a file name, it is quite easy to set up. You only have to specify the DeviceType for the Multimedia control and issue the Open command to open the device. The control takes care of the rest of the task. When there is an Audio CD in your CD drive, the following buttons of the Multimedia control are enabled:

- Prev
- Next
- Play
- Pause
- Stop
- Eject

NOTE The Pause and Stop buttons are only enabled while the CD is playing, and the Play button is only enabled while the CD is stopped.

The sample project for this section is located on the companion CD-ROM as CDPLAYER.VBP.

Setting Up the Basic Program To begin creating a CD player with the Multimedia control, you need to start a new project and then follow these steps:

1. Add the Multimedia control to your Toolbox.
2. Set the `Name` property of the form to `frmCDPlayer` and the `Caption property` to `VB CD Player`.

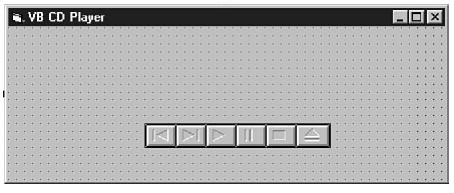
3. Draw the Multimedia control on your form and then set the `Name` property to `mmcCDPlayer`.
4. Set the `DeviceType` property of the control to `CDAudio`.
5. Place the following statement in the `Load` event of the form:

```
mmcCDPlayer.Command = "Open"
```

6. Because several buttons are not used when playing CDs, set the following properties to `False`: `BackVisible`, `StepVisible`, and `RecordVisible`.

When you have completed these steps, your form should look like the one in Figure B3.8.

FIG. B3.8
A simple CD Player can easily be created with the Multimedia control.

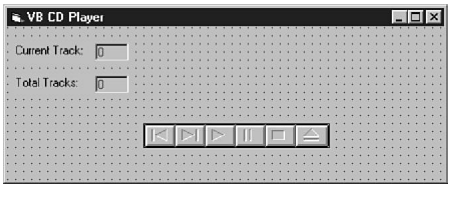


To play a CD, run the program. If you already have an audio CD in the drive, the Play button of the Multimedia control will be enabled and you can start the playback. If there is a data disk or no disk in the drive, only the Eject button will be enabled. The control keeps polling the CD drive to determine when it contains an audio CD. As soon as you place an audio CD in the drive, the appropriate buttons will be enabled.

Determining the Track Number and Total Tracks While you can now play CDs with your program, the program is not very informative to the user. There is no information about the number of tracks on the CD or which track is being played. You can remedy this situation with a couple of additional controls and a little bit of code. You first need to add controls to keep up with the number of tracks and the current track. You can do this with labels, text boxes, or with a slider control. For the sample project, labels were used.

You need to add four Label controls to your form—two to identify the information and two to hold the actual information. The sample application uses two control arrays—`lblCDPlayer` and `lblTracks`—to identify and hold the information, respectively. The updated form is shown in Figure B3.9.

FIG. B3.9
Keep up with the currently playing track of a CD.



To handle placing the information in the labels, you need to add code to the StatusUpdate event of the Multimedia control. The code accesses the Track and Tracks properties of the control. The Track property tells you the current track that is playing, and the Tracks property tells you the total number of tracks on the CD. The following code handles displaying the current track and total tracks:

```
lblTracks(0).Caption = mmcCDPlayer.Track
lblTracks(1).Caption = mmcCDPlayer.Tracks
```

Determining Track Times In addition to determining the track information, you probably want your program to show the length of the current track and of the CD. You also probably want to show the elapsed time on the CD and the elapsed or remaining time for the track. Displaying this information is also handled by using code in the StatusUpdate event to access properties of the Multimedia control. To handle time information, there are four properties that you need to access:

- Length Returns the total length of the CD
- Position Returns the current position of the CD
- TrackLength Returns the length of the current track
- TrackPosition Returns the starting position of the current track

Each of these properties return a time value based on the format specified in the TimeFormat property of the Multimedia control. Although there are a variety of formats that can be used, the simplest time format to work with returns the time values in milliseconds. You need to specify this format by setting the TimeFormat property to 0 in the Load event of your form. You also need a routine that converts the times to an hours:minutes:seconds format to display to the users. This function is shown in Listing B3.4.

NOTE Not all TimeFormat settings are supported by all multimedia devices.

Listing B3.4 FRMCDPLAYER.FRM—Using a Function to Convert Milliseconds into a Readable Format

```
Private Function ConvTime(ByVal TimeIn As Long) As String
  Dim ConvHrs As Integer, ConvMns As Integer, ConvSec As Integer
  Dim RemTime As Long, RetTime As String
  RemTime = TimeIn / 1000
  ConvHrs = Int(RemTime / 3600)
  RemTime = RemTime Mod 3600
  ConvMns = Int(RemTime / 60)
  RemTime = RemTime Mod 60
  ConvSec = RemTime
  If ConvHrs > 0 Then
     RetTime = Trim(Str(ConvHrs)) & ":"
  Else
```

continues

Listing B3.4 Continued

```
    RetTime = ""
  End If
  If ConvMns >= 10 Then
    RetTime = RetTime & Trim(Str(ConvMns))
  ElseIf ConvMns > 0 Then
    RetTime = RetTime & "0" & Trim(Str(ConvMns))
  Else
    RetTime = RetTime & "00"
  End If
  RetTime = RetTime & ":"
  If ConvSec >= 10 Then
    RetTime = RetTime & Trim(Str(ConvSec))
  ElseIf ConvSec > 0 Then
    RetTime = RetTime & "0" & Trim(Str(ConvSec))
  Else
    RetTime = RetTime & "00"
  End If
  ConvTime = RetTime

End Function
```

To display the desired times, you need to add a few more Label controls to your form. You need four more controls to identify the numbers and then two controls for the track times and two for the CD times. The final appearance of the form is shown in Figure B3.10.

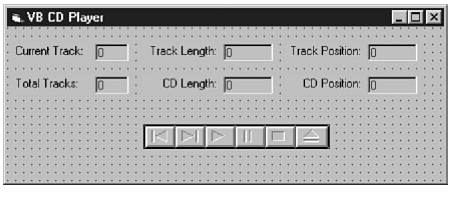


FIG. B3.10
CD Player with track times shown.

Retrieving the track length, CD length, and CD position are quite easy; you simply retrieve the values of the `TrackLength`, `Length`, and `Position` properties respectively and convert the values to the desired time format. To get the position within the track, you have to subtract the starting position of the track (specified by the `TrackPosition` property) from the current CD position (`Position` property). This value is then converted to the appropriate time format. All of this code is placed in the `StatusUpdate` event so that the information gets updated on a regular basis. Listing B3.5 shows the final code for the `StatusUpdate` event.

Listing B3.5 FRMCDPLAYER.FRM—Updating Times from the *StatusUpdate* Event

```
Private Sub mmcCDPlayer_StatusUpdate()
  lblTracks(0).Caption = mmcCDPlayer.Track
  lblTracks(1).Caption = mmcCDPlayer.Tracks
  lblTrackTime(0).Caption = ConvTime(mmcCDPlayer.TrackLength)
```

```
  lblTrackTime(1).Caption = ConvTime(mmcCDPlayer.Position_
       - mmcCDPlayer.TrackPosition)
  lblCDTime(0).Caption = ConvTime(mmcCDPlayer.Length)
  lblCDTime(1).Caption = ConvTime(mmcCDPlayer.Position)
End Sub
```

Running Movies

The Multimedia control also makes it easy for you to run AVI movies from your applications. These movies are stored in AVI files on your computer. The basic setup for running movies is very similar to that which was used for setting up the CD application. The only additional setup required is that the name of a file containing an AVI clip must be entered into the `FileName` property. After a valid file is opened, the Multimedia control sets the enabled properties of the buttons for the movie. All buttons except the Record and Eject buttons will be enabled.

The sample project for playing AVI movies is on the companion CD as AVIPLAYER.VBP.

To create a sample AVIPlayer program, perform the following steps:

1. Start a new project and make sure the `Multimedia` control is part of your Toolbox. Then set the Name of your form to `frmAVIPlayer` and the Caption to `AVI Movie Player`.

2. Place a `Multimedia` control on your form and name it `mmcAVIPlayer`. Set the `RecordVisible` and `EjectVisible` properties of the control to `False` to hide the unused buttons.

3. Add a CommonDialog control to the form to facilitate retrieving file names. (The CommonDialog control is a custom control that can be added from the Project Components menu.) Change the name of the control to `cdlGetFile`.

   ▶ **See** "Using Built-In Dialog Boxes," in Chapter 6.

4. Add a command button to the form to allow the user to start the process of selecting and opening an AVI file. Name the command button `cmdAVI` and set its Caption to `Get and Play AVI Movie`.

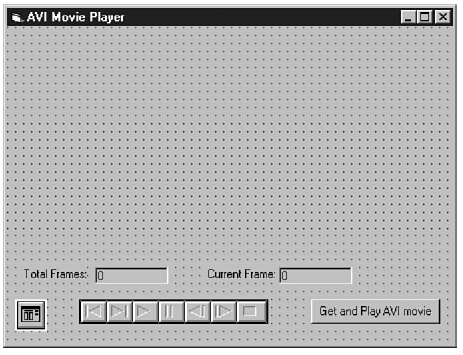
5. Place `Label` controls on the form to identify and hold the Length and Position information for the movie. At this point, your form should look like the one in Figure B3.11.
6. Place the code in Listing B3.6 in the `Click` event of the command button. This code uses the common dialog to retrieve the file and then sets up the `Multimedia` control.

Listing B3.6 FRMAVIPLAYER.FRM—Retrieving and Opening an AVI File

```
Private Sub cmdAvi_Click()
  cdlGetFile.Filter = "AVI Movie Files (*.avi)|*.avi"
  cdlGetFile.ShowOpen

  mmcAVIPlayer.DeviceType = "AVIVideo"
  mmcAVIPlayer.filename = cdlGetFile.filename
  mmcAVIPlayer.Command = "Open"
  lblStatus(0).Caption = mmcAVIPlayer.Length
End Sub
```

FIG. B3.11
The basic setup for playing AVI movies.



One final piece of code that you need is a line to update the current position of the movie. The following line of code should be placed in the `StatusUpdate` event of the Multimedia control:

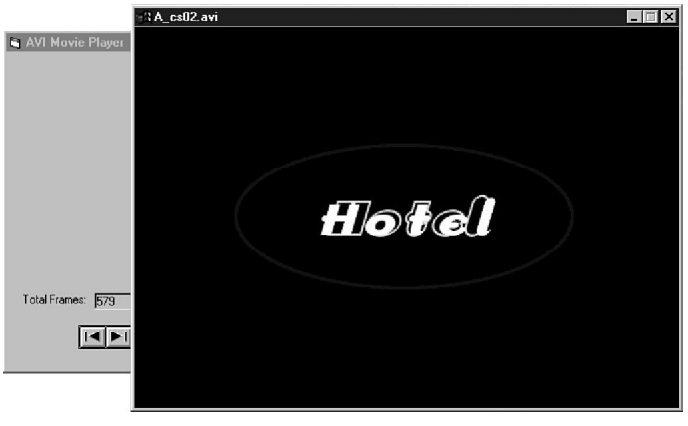
```
lblStatus(1).Caption = mmcAVIPlayer.Position
```

NOTE The length and position of an AVI movie is specified in frames instead of a time.

You can now run your program and start playing movies. After the program has started, you can click the command button to bring up the dialog box to open a file. After you have specified a valid file name and opened the file, the Multimedia control enables the appropriate buttons. You can press the Play button to start the movie. When the movie starts playing, it is shown in a separate window from the one that contains your Multimedia control. Figure B3.12 shows an AVI movie from a Windows help video being run in a separate window.

NOTE There are a number of AVI movies in the VBOnline folder of your Visual Basic installation.

FIG. B3.12
Movies typically run in their own window.

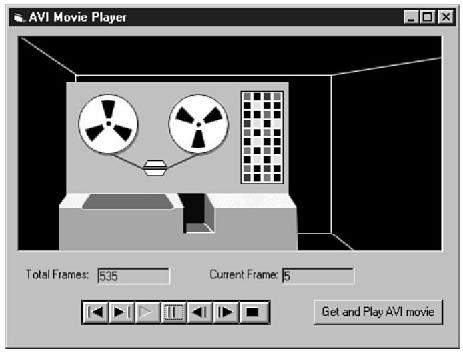


If you want the movie to run on the same form as your control, you can set the `hWndDisplay` property of the Multimedia control to the `hWnd` property of the form. This causes the movie to run in the background of the form. However, running the movie this way can cause problems with the display of controls such as labels. A better solution is to place a PictureBox control on the form and use it to display the movie. The following line of code causes a movie to be run inside a picture box on the form:

```
mmcAVIPlayer.hWndDisplay = picMovie.hWnd
```

Figure B3.13 shows the movie running inside the PictureBox control.

FIG. B3.13
Run movies in a picture box to better control their location.

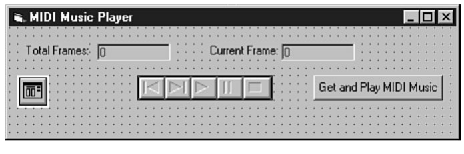


Working with MIDI Files

The setup for a MIDI file player is almost identical to that of the AVI movie player. Your program needs a way to specify and retrieve a file, and then to open the file and start playing the MIDI music. When a valid MIDI file has been opened, the Prev, Next, Play, Pause, and Stop buttons of the Multimedia control are enabled.

The key difference between the player for MIDI files and the player for AVI files is the setting of the `DeviceType` property of the Multimedia control and the `Filter` property of the CommonDialog control. For MIDI files, you need to set the `DeviceType` property to `Sequencer`. This defines the MIDI device to the Multimedia control. Figure B3.14 shows the MIDI player application and Listing B3.7 shows the code to make it run.

FIG. B3.14
A MIDI player created with the Multimedia control.



Listing B3.7 MIDIPLAYER.FRM—Setting the *DeviceType* to *Sequencer* for MIDI Files

```
Private Sub cmdMIDI_Click()
  cdlGetFile.Filter = "MIDI Files (*.mid)|*.mid"
  cdlGetFile.ShowOpen

  mmcMIDIPlayer.DeviceType = "Sequencer"
  mmcMIDIPlayer.filename = cdlGetFile.filename
  mmcMIDIPlayer.Command = "Open"
  lblStatus(0).Caption = mmcMIDIPlayer.Length
End Sub
```

NOTE A couple of MIDI files come with Windows 95 and are located in the \Windows\Media directory. ■

Playing WaveForm Audio

You can use WaveForm Audio (WAV files) in your programs for sound effects and alerts. For example, I have a VB program that reads caller identification information from the phone company and then plays a WAV file depending on who the caller is. The easiest way to play WAV files from a VB application is to use the Windows API function sndPlaySound. The Windows API is a group of Windows functions that you can call from Visual Basic.

To use sndPlaySound, do the following:

1. Start a new Standard EXE project.
2. Add a new code module to the project.
3. In the General Declarations section of the code module, add the following lines:

   ```
   Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA"
        (ByVal lpszSoundName As String, ByVal uFlags As Long) As Long
   Public Const SND_ASYNC = &H1
   Public Const SND_SYNC = &H0
   Public Const SND_LOOP = &H8
   ```

   The first line of code is the *API declaration* for sndPlaySound. It is similar to the first line of a user-defined function in that it lists the function name and required parameters, but the actual function code is in a separate DLL file. The next few lines are constant values used with the function.

4. Call the sndPlaySound function from your program.

   ```
   Private Sub cmdPlaySound_Click()
       Dim lRetVal As Long
       lRetVal = sndPlaySound("C:\WAV\MySound.WAV", SND_ASYNC)
       MsgBox "Done"
   End Sub
   ```

As you can see from the code example, the sndPlaySound function is fairly simple. It takes two parameters, the name of the WAV file and a constant, and returns a value of type long.

The Flags parameter is used to modify the behavior of sndPlaySound. In the preceding code example, the SND_ASYNC constant indicates that the sound is played *asynchronously* from the rest of the program. In other words, the MsgBox statement is immediately executed. If the SND_SYNC constant is used, the message box does not appear until after the sound finishes playing. The SND_LOOP constant can be combined with SND_ASYNC to create a continuous background sound, as in the following line of code.

```
lRetVal = sndPlaySound("C:\WAV\MySound.WAV", SND_ASYNC + SND_LOOP)
```

The sound continues playing until the program executes another sndPlaySound call.

From Here...

This chapter has provided you with an introduction to the world of multimedia programming. You have seen how you can add animation effects to your programs and how you can use the multimedia devices of your computer to play music, show movies, play audio CDs, and add sound effects or narrations to your programs. This chapter also touched on a number of other topics that are covered elsewhere in this book. For more information about these topics, refer to the following chapters:

- To learn more about using the CommonDialog control in your programs, see Chapter 6, "Using Dialogs to Get Information."
- To learn more about text effects for enhancing your applications, see Chapter 14, "Working with Text, Fonts, and Colors."
- To learn more about accessing the API functions of Windows, see Chapter 46, "Accessing the External Functions: The Windows API."
- To learn more about handling graphics images, including the creation of dynamic charts, see "Doing Graphics" on the CD-ROM.