Build Multimedia SoundTracks With WaveMix API

Click & Retrieve Source CODE!

Use the WaveMix API for harmonious mixing of multiple WAV file audio tracks.

by Mark Pruett

In a Control Room application, the operator is alerted to a potential problem by a repeating alarm bell. At the same time, a digitized human voice calmly repeats "Low Water alarm in Number Two Boiler... Low Water alarm ...." In a museum, a touch-screen kiosk program features the digitized voice of a narrator, explaining the histories of antique musical instruments. The user presses a button on the screen next to a picture of a harpsichord, and as the narrator continues uninterrupted, the sounds of the harpsichord can be heard in the background. These are both examples of mixed wave-form audio, the ability to dynamically layer several audio layers into a single sound track. You can do this today using the WaveMix dynamic link library.

Microsoft Windows has supported the playback of wave-form audio, or WAV files, since Windows 3.0. This ability became a standard feature of Windows 3.1. To play back WAV files, the computer must be equipped with a compatible sound card and Windows sound drivers. Most multimedia computers sold today come with these prerequisites installed.

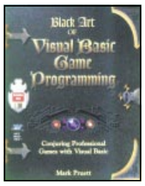
A simple call to the Windows API function, sndPlaySound, is all that is needed to play a WAV file. But Windows' innate ability to play wave-form audio is limited to one WAV file at a time. Attempting to play a second WAV file silences the first WAV file.

Microsoft overcame this dilemma when it released Microsoft Arcade for Windows 3.1. To handle the multiple layered sounds it needed for this set of classic arcade games, Microsoft developed a 16-bit set of routines called WaveMix. This set of routines was later documented and released to developers in Microsoft's Multimedia JumpStart CD and as part of the Microsoft Developer's Library CD. Since then, a 32-bit version of this tool, WavMix32, has been released. There's little difference between the two versions, except for the 32-bit size of many of the data elements involved. I'll focus on the 32-bit version here, but the concepts are applicable to the 16-bit version as well.

THE WAVEMIX API

WaveMix has one simple purpose: it provides an easy way for the programmer to create layered wave-form audio. It requires only two files: WAVEMIX.INI, placed in the \WINDOWS directory, and WAVMIX32.DLL, placed in the \WINDOWS\SYSTEM directory. The DLL is a set of about a dozen function calls.

Mark Pruett is the author of Black Art of Visual Basic Game Programming, *published by Waite Group Press. Reach him on CompuServe at 74133,3406 or on the Internet at pruettm@vancpower.com.*



Conceptually, WaveMix provides a set of eight independent channels. Each of these channels can play a different WAV file. Channels can be looped, so their assigned WAV file plays repeatedly. In my control room example, the alarm bell sound would certainly have been looped so that it would continue to sound automatically until the control room operator shut it off.

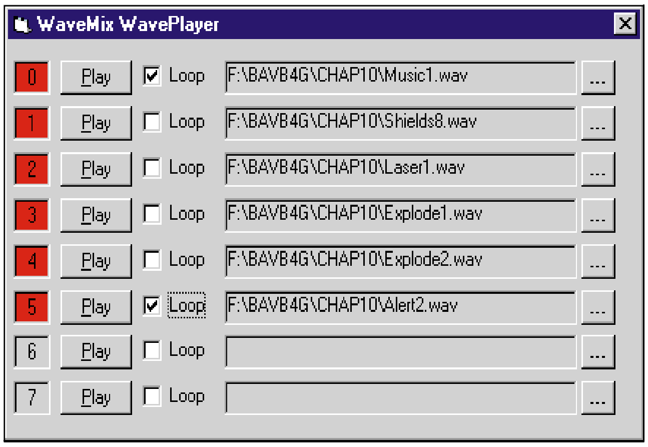
Because each channel is independent, it can be started and stopped at any point during program execution. The sound from one channel doesn't interrupt another channel; it is simply heard together with it, just as the dialogue in a movie can be heard over the background music.

WaveMix is like an engine. It needs to be started up before it's used, and it needs to be shut down when its work is done. Two WaveMix API calls let you fire up the WaveMix DLL. The easiest of the two, the WaveMixInit function, requires no parameters. It returns a Long Integer value called a handle, which will be a required parameter for virtually all of the subsequent WaveMix function calls. WaveMixInit uses values specified in the WAVEMIX.INI file to initialize the WaveMix engine.

The WaveMixConfigInit function gives you more control over how WaveMix starts up. It requires a tMixConfig user-defined type as a parameter. The tMixConfig structure lets you override some of the values in WAVEMIX.INI. In practice, probably the only value you'll want to set is wChannels, which enables WaveMix to play stereo sounds.

WaveMix plays up to eight WAV files simultaneously. Individual WAV files are assigned to channels. Two calls are required to assign a WAV file to a channel: WaveMixOpenWave and WaveMixOpenChannel.

```
WaveFileName = "C:\WINDOWS\DING.WAV"
lpWaveMix = WaveMixOpenWave _
```



0	Play	☑ Loop	F:\BAVB4G\CHAP10\Music1.wav	...
1	Play	☐ Loop	F:\BAVB4G\CHAP10\Shields8.wav	...
2	Play	☐ Loop	F:\BAVB4G\CHAP10\Laser1.wav	...
3	Play	☐ Loop	F:\BAVB4G\CHAP10\Explode1.wav	...
4	Play	☐ Loop	F:\BAVB4G\CHAP10\Explode2.wav	...
5	Play	☑ Loop	F:\BAVB4G\CHAP10\Alert2.wav	...
6	Play	☐ Loop		...
7	Play	☐ Loop		...

FIGURE 1 ***Test Combinations with WavePlayer.*** *Assign different WAV files to the eight available channels, optionally looping selected channels.*

```
  (hMixSession, WaveFileName, 0, 0)
ReturnCode = WaveMixOpenChannel (hMixSession, 5, 0)
```

In the example, WaveMix opens the DING.WAV file and assigns it to channel five. WaveMix's eight channels are numbered zero through seven, and can be assigned in any order. The hMixSession parameter passed to both functions is the WaveMix handle we were assigned when we started up the WaveMix DLL earlier. Always keep track of the audio resources you've opened. In our example, that means keeping track of the value WaveMix assigns to lpWaveMix. You'll need this later to release the Windows resources that WaveMix allocates for each WAV file.

Actually, as I'll show you, the WAV file loaded by WaveMixOpenWave doesn't have to be played on channel five. It can be played on any channel previously opened by a call to

VB4

```
Option Explicit
' WAVEMIX.BAS
' This module contains declarations for all the
' functions in the WaveMix DLL, and provides some
' higher-level functions to simplify using WaveMix
Global hMixSession As Long
Global lpWaveMix() As Long
Global WaveHandle As Long
Global WAVMIX_Quiet As Integer
Global Const WAVEMIX_MAXCHANNELS = 8
Type tChannelInfo
  Loops As Long
  WaveFile As String
End Type
Type tWaveMixInfo
  wSize As Integer
  bVersionMajor As String * 1
  bVersionMinor As String * 1
  szDate(12) As String
  dwFormats As Long
End Type
Type tMixConfig
  wSize As Integer
  dwFlagsLo As Integer
  dwFlagsHi As Integer
  wChannels As Integer
  wSamplingRate As Integer
End Type
Private Type tMixPlayParams
  wSize          As Integer
  hMixSessionLo     As Integer
  hMixSessionHi     As Integer
  iChannelLo     As Integer
  iChannelHi     As Integer
  lpMixWaveLo    As Integer
  lpMixWaveHi    As Integer
  hWndNotifyLo      As Integer
  hWndNotifyHi      As Integer
  dwFlagsLo         As Integer
  dwFlagsHi         As Integer
  wLoops            As Integer
End Type

Declare Function WaveMixInit Lib _
  "WAVMIX32.DLL" ()As Long
Declare Function WaveMixConfigureInit Lib _
  "WAVMIX32.DLL" (lpConfig As tMixConfig) As Long
Declare Function WaveMixActivate Lib "WAVMIX32.DLL" _
  (ByVal hMixSession As Long, ByVal fActivate As _
  Integer) As Long
Declare Function WaveMixOpenWave Lib "WAVMIX32.DLL" _
  (ByVal hMixSession As Long, ByVal szWaveFilename _
  As String, ByVal hInst As Long, ByVal dwFlags _
  As Long) As Long
Declare Function WaveMixOpenChannel Lib
"WAVMIX32.DLL" _
  (ByVal hMixSession As Long, ByVal iChannel As Long, _
  ByVal dwFlags As Long) As Long
Declare Function WaveMixPlay Lib "WAVMIX32.DLL" _
  (lpMixPlayParams As Any) As Integer
Declare Function WaveMixFlushChannel Lib _
  "WAVMIX32.DLL" _
  (ByVal hMixSession As Long, ByVal iChannel As _
  Integer, ByVal dwFlags As Long) As Integer
Declare Function WaveMixCloseChannel Lib
  "WAVMIX32.DLL" _
  (ByVal hMixSession As Long, ByVal iChannel As _
  Integer, ByVal dwFlags As Long) As Integer
Declare Function WaveMixFreeWave Lib "WAVMIX32.DLL" _
  (ByVal hMixSession As Long, _
  ByVal lpMixWave As Long) _
  As Integer
Declare Function WaveMixCloseSession Lib _
  "WAVMIX32.DLL"(ByVal hMixSession As Long) _
  As Integer
Declare Sub WaveMixPump Lib "WAVMIX32.DLL" ()
Declare Function WaveMixGetInfo Lib "WAVMIX32.DLL" _
  (lpWaveMixInfo As tWaveMixInfo) As Integer
' These two functions are needed to reverse the word
' ordering of fields in the WaveMixPlay function.
Private Function HiWord(ByVal l As Long) As Integer
  l = l \ &H10000
  HiWord = Val("&H" & Hex$(l))
End Function

Private Function LoWord(ByVal l As Long) As Integer
  l = l And &HFFFF&
  LoWord = Val("&H" & Hex$(l))
End Function

Sub WAVMIX_SetFile(FileName As String, AChannel As
Long)
' Assign a new wave file, FileName, to the specified
' channel, AChannel.  If this channel is currently
' assigned another wave file, stop playing the channel
' and free the active wave file.
Dim rc As Long
  If WAVMIX_Quiet Then Exit Sub
  If AChannel > UBound(lpWaveMix) Then
     ReDim Preserve lpWaveMix(AChannel)
     WaveHandle = AChannel
  End If
  ' If another wave is currently assigned to this
  ' channel, free it.
  If lpWaveMix(AChannel) <> 0 Then
     WAVMIX_StopChannel AChannel
     rc = WaveMixFreeWave(hMixSession, _
        lpWaveMix(AChannel))
  End If
  ' Open the new wave and assign it to this channel.
  lpWaveMix(AChannel) = WaveMixOpenWave(hMixSession, _
     FileName, 0, 0)
  rc = WaveMixOpenChannel(hMixSession, AChannel, 0)
End Sub
Sub WAVMIX_Close()
' Stop playing all channels and free all wave files,
' then close down this WaveMix session.
Dim rc As Long
Dim i As Integer
  If WAVMIX_Quiet Then Exit Sub
  If (hMixSession <> 0) Then
     For i = 0 To UBound(lpWaveMix)
        If lpWaveMix(i) <> 0 Then
           WAVMIX_StopChannel CLng(i)
           rc = WaveMixFreeWave(hMixSession, _
              lpWaveMix(i))
        End If
     Next
```

CONTINUED ON PAGE 92.

LISTING 1 ***Starting to Hide Complexity.*** *This BAS module was developed for the 16-bit version of WaveMix. The functions hide some of the more complex aspects of the WaveMix DLL.*

WaveMixOpenChannel. I've made it a habit to always call these two functions in tandem. That way, I always know that I have an available channel for any loaded WAV file.

To reassign a channel you've used before, call the WaveMixFreeWave function:

```
ReturnCode = WaveMixFreeWave(hMixSession, lpWave)
```

VB4

CONTINUED FROM PAGE 90.

```
    rc = WaveMixCloseSession(hMixSession)
    hMixSession = 0
  End If
End Sub
Function WAVMIX_InitMixer() As Integer
' Initialize and activate the WaveMix DLL.
Dim rc As Long
Dim config As tMixConfig
  If WAVMIX_Quiet Then Exit Function
  WaveHandle = 0
  ReDim lpWaveMix(0)
  ChDir App.Path
  config.wSize = Len(config)
  config.dwFlagsHi = 1
  config.dwFlagsLo = 0
  'Allow stereo sound
  config.wChannels = 2
  hMixSession = WaveMixConfigureInit(config)
  rc = WaveMixActivate(hMixSession, True)
  If (rc <> 0) Then
    WAVMIX_InitMixer = False
    Call WaveMixCloseSession(hMixSession)
    hMixSession = 0
  Else
    WAVMIX_InitMixer = True
  End If
End Function
Sub WAVMIX_StopChannel(ByVal ChannelNum As Long)
' Stop playing the specified channel.
Dim rc As Integer
  If WAVMIX_Quiet Then Exit Sub
  If (hMixSession = 0) Then Exit Sub
  rc = WaveMixFlushChannel(hMixSession, ChannelNum, 0)
End Sub
Sub WAVMIX_Activate(Activate As Long)
' Activate the WaveMix DLL.
Dim rc As Integer
  If WAVMIX_Quiet Then Exit Sub
  If (hMixSession = 0) Then Exit Sub
  rc = WaveMixActivate(hMixSession, Activate)
End Sub
Sub WAVMIX_PlayChannel(ChannelNum As Long, _
  LoopWave As Long)
' Play a specified channel, and indicate whether the
' sound should be looped.
Dim params As tMixPlayParams
Dim rc As Long
  If WAVMIX_Quiet Then Exit Sub
  If ChannelNum > UBound(lpWaveMix) Then Exit Sub
  If (hMixSession = 0) Then Exit Sub
  params.wSize = Len(params)
  params.hMixSessionLo = LoWord(hMixSession)
  params.hMixSessionHi = HiWord(hMixSession)
  params.iChannelLo = LoWord(ChannelNum)
  params.iChannelHi = HiWord(ChannelNum)
  params.lpMixWaveLo = LoWord(lpWaveMix(ChannelNum))
  params.lpMixWaveHi = HiWord(lpWaveMix(ChannelNum))
  params.hWndNotifyLo = 0
  params.hWndNotifyHi = 0
  params.dwFlagsHi = 5
  params.dwFlagsLo = 0
  params.wLoops = LoopWave
  rc = WaveMixPlay(params)
End Sub
```

After that, just call WaveMixOpenWave and WaveMix-OpenChannel again, this time assigning your new WAV file.

MAKING NOISE

Now that you've assigned all your waves to WaveMix channels, you're ready to play a channel. The good news is that a single call, WaveMixPlay, handles this task. The bad news is that it requires a rather Byzantine user-defined type as a parameter.

Actually, it's not that bad. WaveMixPlay takes tMixPlayParams as a parameter, and this data structure must be filled with a lot of necessary information:

```
Dim MixPlay As tMixPlayParams
Dim ChannelNum as Long
Dim rc As Long

ChannelNum = 5
MixPlay.wSize = Len(MixPlay)
MixPlay.hMixSessionLo = LoWord(hMixSession)
MixPlay.hMixSessionHi = HiWord(hMixSession)
MixPlay.iChannelLo = LoWord(ChannelNum)
MixPlay.iChannelHi = HiWord(ChannelNum)
MixPlay.lpMixWaveLo = LoWord(lpWaveMix)
MixPlay.lpMixWaveHi = HiWord(lpWaveMix)
MixPlay.hWndNotifyLo = 0
MixPlay.hWndNotifyHi = 0
MixPlay.dwFlagsHi = 5
MixPlay.dwFlagsLo = 0
MixPlay.wLoops = LoopWave
rc = WaveMixPlay(MixPlay)
```

Because of the word ordering required by this user-defined type, the WaveMix handle, the value assigned by WaveMixOpenWave, and several other fields must have their high and low 16-bits swapped. Once all the word-swapping nonsense is done, the only field of real interest to us is wLoops. You can loop the audio in a channel so that after the WAV file is played, it can automatically restart at its beginning. Set wLoops to the number of times you want the WAV file to repeat. Set wLoops to -1 and the WAV will loop until you tell it to stop. Looping is an excellent way to take a relatively short piece of music and create a continuous background track.

To stop the sound in a channel, just call Wave-MixFlushChannel:

```
rc  = WaveMixFlushChannel (hMixSession, ChannelNum, 0)
```

Again, hMixSession is the handle returned when you initialized WaveMix, and ChannelNum is the channel you want to shut off.

When your program is finished using WaveMix, it absolutely must shut it down properly. Luckily, this is easy to do. First, call WaveMixFreeWave for each WAV file you previously loaded using WaveMixOpenWave. Then call WaveMixCloseSession, passing it the hMixSession handle assigned when WaveMix was initialized. Now WaveMix is properly shut down.

While using the WaveMix API isn't all that difficult, there's way too much "bookkeeping" involved. You need to worry about starting WaveMix, setting up the user-defined types properly, releasing Windows resources, and shutting down the thing when you're done.

I first started using WaveMix in its 16-bit incarnation. I decided to encapsulate some of the boring tasks into a Visual Basic 3.0 BAS module, which made programming a lot easier. When I moved to 32-bit WaveMix, I ported that module into the

VB4 world (see Listing 1).

Even though I had hidden much of the drudge work inside WAVMIX.BAS, I hadn't reached the level of encapsulation I wanted. With the advent of classes in VB4, I could now take WaveMix a step further: I could create a WaveMix object.

The cWaveMix class module describes that object. Rather than rewriting all my code from scratch, cWaveMix is a wrapper around my WAVEMIX.BAS functions.

WHY A CLASS?

Why bother building a class? One big reason is to make use of the class' built-in Initialize and Terminate events. Because these events are fired automatically, I can place my WaveMix start-up and shut-down code in these events, and be guaranteed they'll be called properly. Using my cWaveMix class, the programmer no longer needs to remember to call these

VB4

```
Option Explicit
' cWaveMix Class - encapsulates the WavMix32 DLL.
Dim Channel(0 To 7) As tChannelInfo
Public CurrentChannel As Long
' The Initialize Event - makes sure the WaveMix DLL is
' started properly.
Private Sub Class_Initialize()
  If Not WAVMIX_InitMixer() Then
     MsgBox "Unable to Initialize WaveMix DLL", _
        vbOKOnly Or vbExclamation, _
        "WaveMix Error"
     End
  End If
End Sub
' The Terminate Event- makes sure the WaveMix DLL is
' shut down properly.
Private Sub Class_Terminate()
  WAVMIX_Close
End Sub
' The Play Method-starts playing the wave file
' associated with the current channel.
Public Sub Play()
  WAVMIX_PlayChannel CurrentChannel, _
     Channel(CurrentChannel).Loops
End Sub
' The FileName Property - associates the current
channel
' with a particular wave file.
Public Property Get FileName() As String
  FileName = Channel(CurrentChannel).WaveFile
End Property
Public Property Let FileName(ByVal AFileName As String)
  Channel(CurrentChannel).WaveFile = AFileName
  WAVMIX_SetFile Channel(CurrentChannel).WaveFile, _
     CurrentChannel
End Property
' The Quit Method - Stops the current channel if it's
' playing.
Public Sub Quit()
  WAVMIX_StopChannel CurrentChannel
End Sub
' The AutoLoop Property - determines if the current
' channel will loop its wave file.
Public Property Get AutoLoop()
  AutoLoop = Channel(CurrentChannel).Loops
End Property

Public Property Let AutoLoop(LoopWave)
  Channel(CurrentChannel).Loops = LoopWave
End Property
```

LISTING 2 ***A WaveMix Class.*** *This VB4 class module is a wrapper around WAVMIX.BAS. It hides virtually all of the internals of the WaveMix DLL, exposing just three properties and two methods.*

routines. In fact, the programmer doesn't even need to know about them.

The cWaveMix class is very simple, consisting of only three properties and two methods (see Listing 2). In designing the interface to this class, I wanted to make it as simple and as foolproof as possible. There are easily a half-dozen alternatives one might take in designing this class. I chose the one that worked best for me.

Remember that eight channels are available for playing sounds. I considered presenting an array of some sort to the programmer, but that direction never seemed clean enough. I finally chose a selector property called CurrentChannel which you set to a value from zero to 7. After that, any other properties that are set, or methods that are called, are effective for that particular channel.

The other two properties are FileName and AutoLoop. Two methods, Play and Quit, start and stop the sounds playing in the current channel. A simple example illustrates how these are used:

```
WaveMix.CurrentChannel = 5
WaveMix.FileName = "C:\WINDOWS\DING.WAV"
WaveMix.AutoLoop = 0
WaveMix.Play
```

Here, I assign the wave-form audio file DING.WAV to channel five and play it one time. This is a lot easier than calling all those API functions.

Here's an interesting aside: I implemented a Quit method to let the programmer stop a currently playing channel. I had originally wanted to call it the Stop method because this was more consistent with terminology used by the MCI Multimedia control. Unfortunately VB4 didn't care much for my idea because the name Stop conflicts with Visual Basic's own Stop statement, used to suspend program execution. I find this more than a little bewildering. I'd expect VB to be able to distinguish between WaveMix.Stop and Stop, because WaveMix.Stop is unambiguous. But it can't, so I decided to just call my method Quit. Hey, I'm flexible.

To tie all the pieces of this WaveMix journey together, I wrote an example program that allows you to play with most of WaveMix's features. WavePlayer lets you assign a different WAV file to each channel, and lets you optionally loop each channel (see Figure 1).

You can get the source code for the example program, as well as all the code shown here, from *VBPJ's* online sites. The example program, contained in the file ID0495.ZIP, is short because all the messy details of the WaveMix DLL are hidden inside the cWaveMix class. Just add WAVMIX.BAS and CWAVEMIX.CLS to your projects and take advantage of eight-channel stereo wave-form audio. Download ID0495.ZIP from the *VBPJ* Development Exchange on the World Wide Web (http://www.windx.com), or from *VBPJ's* CompuServe Forum or MSN site. For details, see "How to Reach Us" in *VBPJ's* Letters to the Editor.

Finally, Microsoft recently released its Game SDK, also known as DirectX, designed to make game and multimedia programming easier. Part of the SDK is DirectSound, a set of routines that provide low-level access to sound devices. Unfortunately for VB programmers, DirectX is designed as a C++ class library. At the very least, much of DirectX will likely need to be wrapped in helper DLLs or OCXs before it is usable by VB. The advantage of WaveMix is that you can use it in either Windows 3.1 or Windows 95, and you can run it today from VB.