

Introduction to Multimedia Systems

Editors

Gaurav Bhatnagar

Shikha Mehta

Sugata Mitra

Centre for Research in Cognitive Systems (CRCS)

NIIT Ltd.

New Delhi, India



ACADEMIC PRESS

A Harcourt Science and Technology Company

Netflix v. VL Collective

IPR2023-00891

Netflix Ex. 1027

San Diego San Francisco New York Boston London Sydney Tokyo

Cover art: © 2001 John Foxx Images

This book is printed on acid-free paper. ∞

Copyright © 2002 by ACADEMIC PRESS

All Rights Reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

Requests for permission to make copies of any part of the work should be mailed to:
Permissions Department, Harcourt Inc., 6277 Sea Harbor Drive,
Orlando, Florida 32887-6777

Explicit permission from Academic Press is not required to reproduce a maximum of two figures or tables from an Academic Press chapter in another scientific or research publication provided that the material has not been credited to another source and that full credit to the Academic Press chapter is given.

Academic Press

A division of Harcourt, Inc.

525 B Street, Suite 1900, San Diego, California 92101-4495, USA

<http://www.academicpress.com>

Academic Press

Harcourt Place, 32 Jamestown Road, London NW1 7BY, UK

<http://www.academicpress.com>

Library of Congress Catalog Card Number: 2001092781

International Standard Book Number: 0-12-500452-4

International Standard Book Number: 0-12-500453-2 (International edition)

PRINTED IN THE UNITED STATES OF AMERICA

01 02 03 04 05 06 MB 9 8 7 6 5 4 3 2 1

The WAV File Format

INTRODUCTION

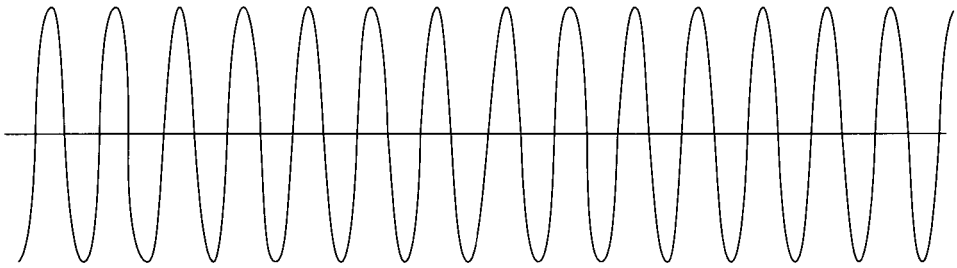
Waveform audio is recognized as one of the essential components of a multimedia system. With waveform audio, you can do anything within the practical limitations of memory, disk capacity, sound card capabilities, and so on. The normal functions that you perform on your tape recorder can also be done with waveform audio.

Like CD Audio, waveform audio is a digital medium. However, unlike CD audio, it supports a variety of formats, from 8-bit mono at a sample rate of 11 Kbps to 16-bit stereo at sample rate of 44 Kbps. Also, unlike CD audio, you can read and write it, store and retrieve it, and manipulate it as you like.

DIGITAL AUDIO BASICS

Sound consists of pressure waves moving through a medium, such as air. For each wave pulse there is a traveling zone of compressed air trailed by a zone of rarefied air. When represented electronically, compression is represented by a positive voltage and rarefaction is represented by a negative voltage. The voltage level determines the amplitude of the wave. A pure sine wave would be gradually switching from a positive to a negative value of the same degree and then back again as shown in Figure 1.

The analog waveform stored digitally has the voltage levels of the wave stored at regular intervals as a set of numbers. Audible frequencies range from 20–20,000 Hz. Therefore, to determine a meaningful digital representation, you need to sample the waveform. Sampling is the process of taking a snapshot of the waveform at regular intervals and representing it as a binary number. This process is called PCM (Pulse Code Modulation). To accurately represent any audio signal, we sample at twice the maximum frequency contained in the wave source. This is the reason why CD audio is recorded at 44.1 kHz—which is twice the maximum audible frequency of humans. You need not, however, always stick to this frequency; you may sample at lower sampling rates.

**FIGURE 1**

A pure sine wave.

Therefore, to record audio at 11 kHz, you need to sample 11,000 times per second.

The format of the data stored is very simple. In 8-bit samples, each byte represents amplitude. If the sample contains two channels of audio, the left channel is recorded in the even-numbered bytes while the right channel is recorded in the odd-numbered bytes. In 8-bit sampling, the baseline of the data is represented by 128. All positive amplitudes have values over 128 while all negative amplitudes have values below 128.

In 16-bit sampling, amplitude values range from -32,768 to 32,767.

THE STRUCTURE OF THE WAVE AUDIO FILE

The wave file not only contains the digital data required to produce sound but also additional information such as the sampling rate, the type of audio data, and other critical data. We can perform tape recorder functions like Play and Pause using the normal MCI (Multimedia functions available in the windows API). However, when it comes to modifying a wave file, we cannot use the MCI functions. Hence, we use the MMIO (MultiMedia Input Output) low-level functions available in the Windows API, for which we need to know the actual structure of the wave file.

Wave data comes in several formats, sampling rates, channels, and resolutions (bits/sample). All we need to know is that the information appears in a format block called the "Resource Interchange File Format," or the RIFF chunk.

Figure 2 represents a file, TADA.WAV which is a standard file packaged with Win95. As the figure suggests, RIFF files have hierarchical structures—chunks which in turn contain more chunks.

The highest level chunk is the RIFF chunk itself. Chunks are identified by labels, known as Chunk Ids. The first four bytes contain the characters RIFF. To read and write RIFF files, the standard data structure called MMCKINFO (MultiMedia ChunK INfOrmation) is used. The structure as defined in Visual Basic is:

- Type MMCKINFO
- Ckid as FOURCC
- Cksize as Long
- FccType as FOURCC
- dwDataOffset as Long
- dwFlags as Long
- End Type

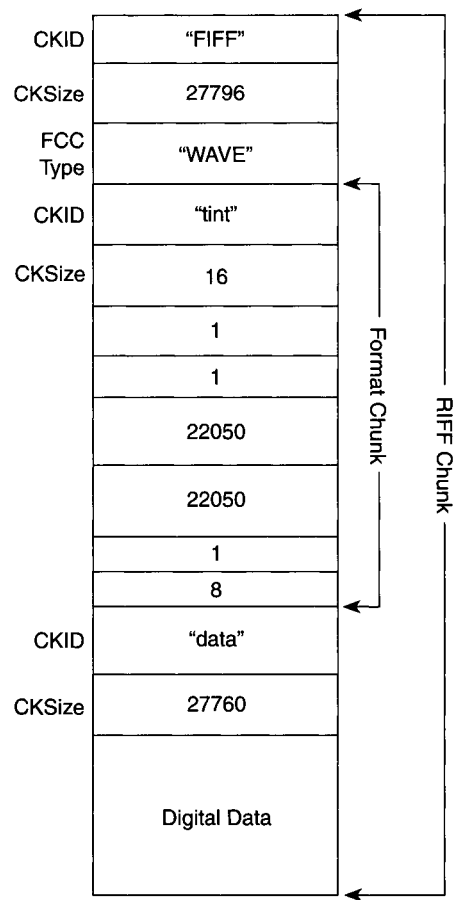


FIGURE 2
Sample waveform structure of TADA.WAV.

where structure FOURCC is defined as:

- Type FOURCC
- Chars as String * 4
- End Type

where:

- FOURCC is a 32-bit code used to identify RIFF chunks, and represented as a sequence of 1 to 4 ASCII alphanumeric characters padded on the right with blank characters.
- Ckid represents the identification of the chunk.
- Cksize specifies the size of the data field of the chunk.
- FccType specifies the form type for "RIFF."
- dwDataOffset specifies the file offset of the beginning of the chunk's data field.
- dwFlags specifies flags giving additional information about the chunk.

To read the WAVE format data the structure is

- Type PCMWAVEFORMAT
- Wf as WAVEFORMAT
- wBitsPerSample as Integer
- End Type

where wBitsPerSample specifies the number of bits per sample, and the structure of WAVEFORMAT is as follows:

- Type WAVEFORMAT
- FormatTag as Integer
- nChannels as Integer
- nSamplesPerSec as Long
- nAvgBytesPerSec as Long
- nBlockAlign as Integer
- End Type

where:

- FormatTag specifies the format types (usually PCM).
- Channels specifies the number of channels, 0 signifies mono and 1 signifies stereo.
- SamplesPerSec specifies the sample rate in samples per second.
- AvgBytesPerSec specifies the required average data transfer rate in bytes/second.
- BlockAlign specifies the block alignment in bytes.

ACCESSING THE WAVE DATA

Why should we use MMIO functions instead of the conventional read/write functions? Well, the answer is simple. Although the structures are the same, they vary in size, which means we cannot read them into or write from them into simple, fixed-length record buffers. To navigate through the wave file we are going to use the following MMIO functions (see Table 1).

We use these functions because we do not need to know the actual size of the chunk. To navigate through the structures, all we have to do is set either the chunk id or the form type (FccType), set a search flag, and then call mmioDescend() or mmioAscend(). The function will search for the next chunk in line and fill in the structure with the details for that particular chunk.

A wave file consists of three chunks—the RIFF chunk, PCMFormat chunk, and the DATA chunk. However, there is another chunk called the LIST chunk, which contains

Table 1 MMIO Functions

Function	Description
mmioOpen()	To open a wave file
mmioClose()	To close a wave file
mmioRead()	To read the chunk data
mmioWrite()	To write the chunk data
mmioDescend()	To navigate through a wave file
mmioAscend()	To navigate through a wave file

details such as copyright and description of the data chunk. Usually it is blank, but in some wave files shipped along with Windows 95, this structure contains some information. Table 2 gives the structure of the TADA.WAV file.

The steps for reading a wave file are as follows:

1. Open the Wave file using the mmioOpen function.
2. Create an instance of the structure MMCKINFO, for example, MMCKINFO Parent.
3. Set Parent.FccType.Chars="WAVE" because the FccType of the RIFF chunk is "Wave".
4. Call the mmioDescend function, passing the structure Parent as one of its parameters and a flag MMIO_FINDRIFF.
The Parent structure will now be filled with the RIFF chunk data, which means that when you actually print out the values of Parent.Ckid you will find that it contains "RIFF" and the value of parent. Cksize will be (filesize-8) bytes.
5. Create another instance of the structure MMCKINFO, say MMCKINFO Child.
6. Set the Child.Ckid.Chars = "fmt" and call mmioDescend function, passing the structure Child as one of its parameters and a flag MMIO_FINDCHUNK. This will take the file pointer to the beginning of the Format chunk.
7. Call the mmioRead function, passing an instance of the structure PCMWAVEFORMAT to it along with the parameter Child.Cksize that will tell the function how many bytes to read.
8. Next, to reach the Data chunk, use the mmioDescend function. However, before doing that ascend out of the Format Chunk. Therefore, call the mmioAscend function passing the structure Child to it. This will position the pointer to the beginning of the Format Chunk.
9. Descend to the Data Chunk by setting the Child.Ckid.Chars = "data" and calling the function mmioDescend passing the Child structure and flag MMIO_FINDCHUNK as its parameters.

Table 2 TADA.WAV File Structure

Offset position	Size in bytes	Contents	Belongs to chunk	Comments
0	4	"RIFF"	RIFF	Ckid, each byte contains one character
4	4	27,796	RIFF	Cksize, equals the file size minus eight bytes ie. (RIFF's ckid and cksize)
8	4	"WAVE"	RIFF	FccType
12	4	"fmt"	PCMFormat	Next Ckid, ? stands for blank
16	4	16	PCMFormat	Cksize, Wave format chunk is 16 bytes
20	2	1	PCMFormat	Indicates a PCM WAVE format
22	2	1	PCMFormat	Number of channels
24	2	22,050	PCMFormat	Sampling rate
28	4	22,050	PCMFormat	Average bytes per second
32	2	1	PCMFormat	Effective bytes per second
34	2	8	PCMFormat	Bits per sample
36	4	"data"	Data	Ckid id of the wave data
40	4	27,760	Data	Cksize, Size of the digital data

10. Use `mmioRead` to read the digital data into a byte type buffer. This can be done easily since the size of the data from `Child.Cksize` is known.
11. You can then manipulate this digital data buffer, such as chop it, add special effects to it, and so on, since it is just an array which contains nothing but numbers.
12. Write the RIFF chunk followed by the FORMAT chunk and the DATA chunk to write the modified file back to the disk.
13. If you have appended some data to the audio data buffer, keep in mind to also modify the respective data in the RIFF structure, such as the `Cksize`.

Projects

1. The program (in Visual Basic) to read and play a single file is given at the Web site of this book. Change the code so that it appends another file, and add a button to append the code. The program files can be found by following the link to [Audio/Projects/WavePlay](#). Read the code carefully. You will need to modify the code accordingly. Here are a few hints:
 - Use the `ReDim Preserve <arrayname>` function to increase the size of the data array.
 - Keep adding the chunk size of the newly read file to the previous chunk size.
2. Make an audio presentation on a topic related to your school. Choose a period based on your favorite music, and use clips as a running theme of the presentation. (Classic rock is your favorite? Choose the sixties and seventies as your period.) Add interviews, music, and voice-overs to convey your ideas or the information that you wish to present.

Prelude to Graphics

Graphics have become an integral part of our lives. Pictures, photographs, and mixtures of the two are everywhere. Graphics can be called “good” or “bad” depending on a number of things. Here are some of the things you can do to find out these differences for yourself:

- Compare pictures in newspapers, magazines, TV, and movies.
- Look at a photograph and try to draw it.
- Look at a real object (for example, a cup of tea) and draw it.
- Make a collage by cutting out pictures from magazines and newspapers.

Table 1 will help you think about graphics.

Table 1 Graphics in Various Media

Things to notice	Newspaper	Magazine	TV	Movie	Drawing
Where do you find pictures with more detail?					
Where do pictures look more “real”?					
Where do things look “solid”?					
What kind of picture makes you “feel” something?					