NNUAL GRAPHICS ISSUE!

#202 JULY 1993

Dr. Dobb's JOURNAL

SOFT
TO

NLEASHING THE POWER OF COMPUTER GRAPHICS

MORPHING

PALETTE MAPPING

IMAGE PROCESSING

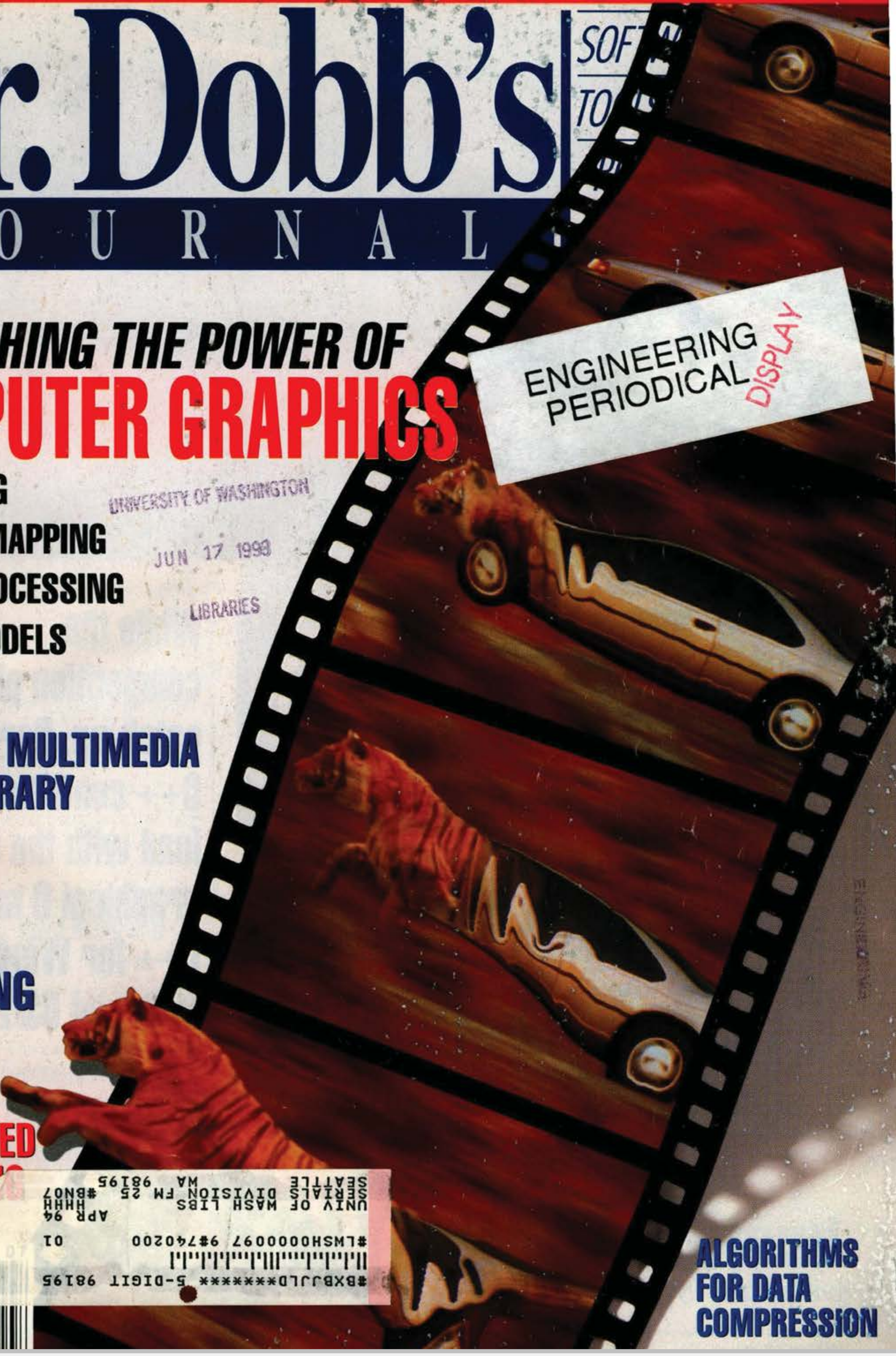
COLOR MODELS

UR OWN
INDOWS MULTIMEDIA
LASS LIBRARY

D-ROM
TENSIONS

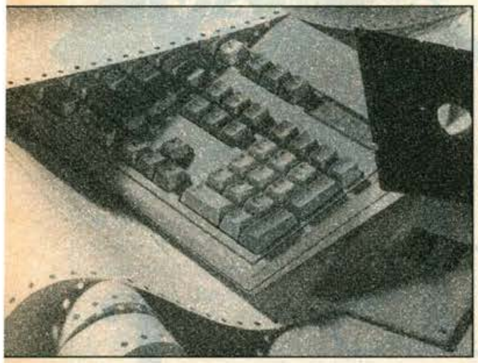
EBUGGING
AL-TIME
OGRAMS

STRIBUTED
MPUTI



ALGORITHMS
FOR DATA
COMPRESSION

($4.95 CANADA)



A Multimedia Class Library for Windows

Encapsulating the Media Control Interface

John Musser

Microsoft Windows' Media Control Interface (MCI) is a standard command language for communicating with multimedia devices—CD, Waveform and MIDI audio, AVI, videodiscs, video overlay devices, audio mixers, and the like. However, even though both Microsoft and Borland provide with their compilers large, comprehensive class libraries—the Microsoft Foundation Class library and ObjectWindows Library, respectively—neither provide object support for multimedia or Windows API multimedia extensions.

This article addresses this gap by showing how to design and implement a comprehensive C++ class library that enhances the MCI interface to multimedia devices. This hierarchy employs encapsulation, inheritance, and polymorphism to create a flexible and extensible framework for controlling multimedia devices under Windows. The result is a set of objects that make programming multimedia easier, more robust, concise, and maintainable. A simple client program, MciMan, demonstrates the use of the Waveform audio and AVI video classes. The class library and client program are compiler independent and can be used with any Windows 3.1 compatible C++ compiler, including Borland's C/C++ and Microsoft's Visual C++. The AVI portions of this code require the digitalv.h *#include* file, Video for Windows drivers, and runtime DLLs that come with the Video for Windows package and its SDK. They're available free of charge from the Windows Extensions forum on CompuServe (GO WINEXT). Vfwrun.zip contains the runtime DLLs and vfwdk.zip is the Video for Windows Development Kit.

John is a senior developer at Personal Media International, a startup specializing in multimedia entertainment software. He can be reached via CompuServe at 70621,1460.

MCI Overview

The MCI specification, released in 1991 by Microsoft and IBM, defines a set of base commands that can be applied to any general device, and extended commands for specific device types. The specification is designed to be extensible so that other devices may be added. For example, the AVI specification was added in 1992. Another example is MediaVision, which supplies an MCI driver that provides audio-mixing capabilities using extended commands specific to this device type. The specification as documented in the Windows SDK identifies 11 device types. Some drivers are supplied with either Windows or the SDK, while others are provided by the device supplier. Table 1 describes most of the currently available MCI drivers. As of this writing, however, not all the device types listed actually had MCI drivers available.

MCI divides all multimedia devices into one of two device types: simple or compound. The basic difference is that sim-

Device Type	Description	Driver Files	Driver Source
animation	Plays Autodesk Animator (.flc/.fli) files	MCIAAP.DRV	Autodesk
	Plays MacroMind Director (.mmm) files	MCIMMP.DRV	MDK, Visual Basic
cdaudio	Controls compact disc audio	MCICDA.DRV	Windows 3.1 SDK, MDK
dat	Controls Digital Audio Tape deck		
digitalvideo	Plays AVI video files	MCIAVI.DRV	Video for Windows
other	An undefined MCI device		
overlay	Controls a video overlay device—analog video in a window	MCIVBLST.DRV	Creative Labs
scanner	Controls an image scanner		
sequencer	Plays MIDI audio files	MCISEQ.DRV	Retail Windows
vcr	Controls a video cassette recorder		
videodisc	Controls the Pioneer LD-V4200 videodisc player	MCIPIONR.DRV	Windows 3.1 SDK, MDK
waveaudio	Plays and records waveform audio files	MCIWAVE.DRV	Retail Windows

***Table 1:** Several Windows MCI devices.*

We slash interface development time across DOS, UNIX, POSIX, VMS...

(and we can prove it!)

C-Programmers:
See for yourself how Vermont Views® can help you create powerful user interfaces—whatever your environment!

If you want to create sophisticated user interfaces—and save *tremendous* time and effort doing it—Vermont Views is exactly what you need.

Vermont Views isn't just a common interface package.

It's a deep, flexible, menu-driven screen designer supported by a C library of over 580 functions. It lets you create the ultimate user interfaces for complex database applications - in a fraction of the time it would take to code it yourself!

With Vermont Views, you create screens interactively. Designing is fast and creative. And changes—both tiny adjustments and huge reworks—are incredibly easy.

Pull down menus, window-based data entry forms with tickertape or memo fields, scrollable regions, choice lists, context sensitive help .... All these interface objects (and more) are immediately accessible. Your DOS applications can have full mouse control, and work in graphics as well as text modes! And with Vermont Views, even terminal-based applications can have the elegant features usually found only on micros.

Create prototypes fast, applications even faster.

With most systems, you have to throw away your prototypes when coding begins. But with Vermont Views, *prototypes become the actual application.*

Menus, data-entry forms, and all screen features are usable in the final applications without change.

NEW! Vermont Views *PLUS*

An enhanced version of Vermont Views for DOS. Vermont Views PLUS provides the interface power needed for developing multi-megabyte, graphic-enhanced C applications. Best of all it supports all popular DOS extenders and graphics libraries. VVPLUS also includes full source code for all libraries.

It's the universal solution.

Vermont Views operates completely independent of hardware, operating system, and database. Any interface you create can be ported easily among DOS, UNIX, XENIX, POSIX, QNX, OS/2, and VMS.

You can use Vermont Views with any database that has a C-language interface (including Oracle, Informix, db_Vista, and C-Tree). You can run on PCs, DEC, NCR, HP, AT&T, and other systems.

No runtime fees or royalties.

Don't take our word for it—put Vermont Views to the test. Call or fax now for your personal, free demonstration kit. Or order Vermont Views with our 60-day, money back guarantee.

Either way, you'll immediately see how Vermont Views can slash your interface development time.

For MS/PCDOS, Vermont Views is $495. Vermont Views PLUS with MemEx, GraphEx and full source code is $795. For UNIX, POSIX, and VMS, prices start at $1,795, depending on machine class.

Vermont Creative Software
1 Pinnacle Meadows, Richford, VT 05476
Phone (802)-848-7731 FAX (802)-848-3502
Please Mention **"Offer** 396"

Call 1-800-848-1248 for a free demo of the ultimate application interface!

CIRCLE NO. 529 ON READER SERVICE CARD

(continued from page 84)

ple devices do not use data files, whereas compound devices usually do. CD audio and videodisc players are simple devices; waveform audio, MIDI sequencers, and AVI are all compound devices.

MCI provides two basic programming interfaces: a command-string interface that allows the use of ordinary text strings such as *play cdaudio* to control multimedia devices (this very open approach is well suited to scripting and authoring applications); and a command-message interface that uses C-language structures and a Windows-style message-passing model for device control. The MCI classes I describe here use the command-message interface because it's more efficient and better suited to general programming.

A single function, *mciSendCommand()*, is used along with a set of "polymorphic" arguments to access the command-message interface. The *mciSendCommand()* function takes:

- A WORD device identifier (analogous to a handle).
- A message-type constant prefixed with MCI_ (such as MCI_PLAY).
- A DWORD value set to one or more flags usually specifying which elements of a given structure contain valid values.
- A far pointer to a data structure containing values to be sent or returned. (The structure is often specific to each message type.)

Nearly all the member functions in this library make at least one call to *mciSendCommand()*. But don't be fooled by its simplicity: It has many options, constants, flags, messages, structures, and return values (and our goal is to hide these).

MCI Class Library

The MCI class hierarchy is designed to encapsulate and enhance the MCI interface and to get the most benefit from the least code. It is part of a C++ library that will be used as part of the basis for the commercial multimedia products we are currently developing. As such, it was designed for a specific immediate purpose, but also had to be capable of handling unknown future requirements. This influenced both the design and the implementation. The design had to be flexible, extensible, and (most of all) practical. Because our needs are for specific devices only, not all MCI devices are directly implemented although adding support for additional devices is a short process. Figure 1 diagrams the MCI class hierarchy, giving an overview of this single-inheritance tree.

All MCI commands are classified into one of four categories: system, required, basic, and extended. Commands specified for the first two types (such as open, close, and status) require support by all device types and are good candidates for member-function definitions in the base class(es) of the MCI class hierarchy. The basic commands, including play, stop, and seek, are supported by most device types and can also be placed at or near the top of the tree. Extended commands that support specific devices can be implemented farther down the hierarchy.

The root node for the MCI class hierarchy, *MCIDevice,* serves as the base class for both the *CompoundDevice* class

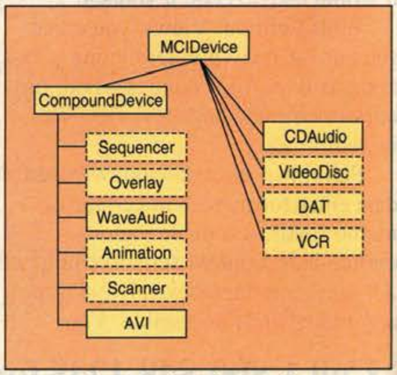


Figure 1: *MCI class hierarchy.*

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.