

# ABOUT FACE

THE ESSENTIALS OF  
USER INTERFACE DESIGN



## ALAN COOPER

**"Father of Visual Basic"**  
**Microsoft Windows Pioneer Award Honoree**

**Foreword by Andrew Singer**

CiM Ex. 1027 Page 1

About Face: The Essentials of User Interface Design  
Published by  
Hungry Minds, Inc.  
909 Third Avenue  
New York, NY 10022  
www.hungryminds.com

Library of Congress Catalog Card No.: 95-75055  
ISBN 1-56884-322-4  
Printed in the United States of America  
20 19 18 16 15 14 13 12 11 10

Distributed in the United States by Hungry Minds, Inc.  
Distributed by CDG Books Canada Inc. for Canada; by Transworld Publishers Limited in the United Kingdom; by IDG Norge Books for Norway; by IDG Sweden Books for Sweden; by IDG Books Australia Publishing Corporation Pty. Ltd. for Australia and New Zealand; by TransQuest Publishers Pre Ltd. for Singapore, Malaysia, Thailand, Indonesia, and Hong Kong; by Gotop Information Inc. for Taiwan; by ICG Muse, Inc. for Japan; by Intersoft for South Africa; by Eyrolles for France; by International Thomson Publishing for Germany, Austria and Switzerland; by Distribuidora Cuspide for Argentina; by LR International for Brazil; by Galileo Libros for Chile; by Ediciones ZETA S.C.R. Ltda. for Peru; by WS Computer Publishing Corporation, Inc., for the Philippines; by Contemporanea de Ediciones for Venezuela; by Express Computer Distributors for the Caribbean and West Indies; by Micronesia Media Distributor, Inc. for Micronesia; by Chips Computadoras S.A. de C.V. for Mexico; by Editorial Norma de Panama S.A. for Panama; by American Bookshops for Finland.

For general information on Hungry Minds' products and services please contact our Customer Care Department within the U.S. at 800-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

For sales inquiries and reseller information, including discounts, premium and bulk quantity sales, and foreign-language translations, please contact our Customer Care Department at 800-434-3422, fax 317-572-4002, or write to Hungry Minds, Inc., Attn: Customer Care Department, 10475 Crosspoint Boulevard, Indianapolis, IN 46256.

For information on licensing foreign or domestic rights, please contact our Sub-Rights Customer Care Department at 212-884-5000.

For information on using Hungry Minds' products and services in the classroom or for ordering examination copies, please contact our Educational Sales Department at 800-434-2086 or fax 317-572-4005.

Please contact our Public Relations Department at 212-884-5163 for press review copies or 212-884-5000 for author interviews and other publicity information or fax 212-884-5400.

For press review copies, author interviews, or other publicity information, please contact our Public Relations department at 317-572-3168 or fax 317-572-4168.

For authorization to photocopy items for corporate, personal, or educational use, please contact Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, or fax 978-750-4470.

**LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY** THE AUTHOR AND PUBLISHER OF THIS BOOK HAVE USED THEIR BEST EFFORTS IN PREPARING THIS BOOK. THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATION OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS BOOK AND SPECIFICALLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE, AND SHALL IN NO EVENT BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGE, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL OR OTHER DAMAGES.

Names and product names used in this book are trademarks, registered trademarks, or service marks of their respective owners. No part of this book is associated with any product or vendor.

About

Alan C  
sultant and  
with a bro  
and success  
approach to  
book.

Since 1976  
(Computer  
programm  
Systems G  
with produ

Bill Gates p  
ference in  
invention o

Alan Coop  
Entrepreneur  
er group in  
er and writ



cursor makes as it merely passes over something on the screen **free cursor hinting**. Once the captive phase has begun, I call changes to the cursor **captive cursor hinting**.

Microsoft Word uses the clever free cursor hint of reversing the angle of the arrow when the cursor is to the left of text to indicate that selection will be line-by-line or paragraph-by-paragraph instead of character-by-character as it normally is within the text itself. Many other programs use a hand-shaped cursor to indicate that the document itself, rather than the information in it, is draggable.

Microsoft is using captive cursor hinting more and more as they discover its usefulness. Dragging-and-dropping text in Word or cells in Excel are accompanied by cursor changes indicating precisely what the action is and whether the objects are being moved or copied. In Windows 95, when you drag a file in the Explorer, you actually drag the text of the name of the file from one place to another.

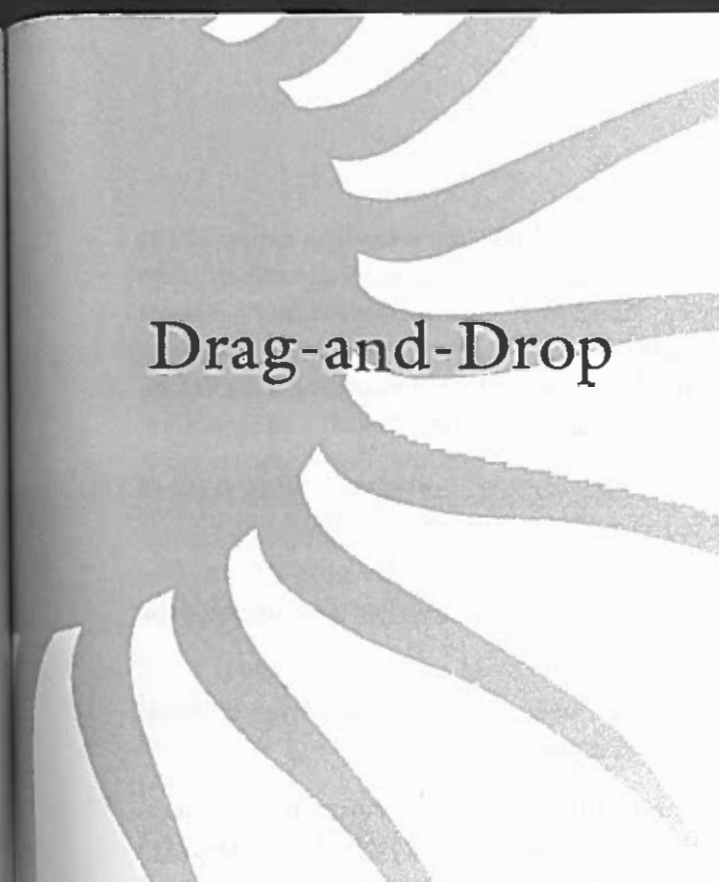
When something is dragged, the cursor must drag either the thing or some simulacrum of that thing. In a drawing program, for example, when you drag a complex visual element from one position to another, it may be too difficult for the program to actually drag the image (due to the computer's performance limitations), so it often just drags an outline of the object. If you are holding down the CTRL key during the drag to drag away a copy of the object instead of the object itself, the cursor may change from an arrow to an arrow with a little plus sign over it to indicate that the operation is a copy rather than a move. This is a clear example of captive cursor hinting.

## Drag-and-Drop

**O**f all the direct-manipulation idioms of the GUI, nothing defines it more than the operation, clicking and holding the button on some object across the screen. Surprisingly, it isn't used as widely as we imagine, and it hasn't lived up to its full potential.

### Whither drag-and-drop?

Any mouse action is very efficient because it combines command components in a single user action: location and a specific function. Drag-and-drop is efficient because, in a single, smooth action, it moves an object to a new geographical location. Although drag-and-drop is immediately as a cornerstone of the modern GUI, it is remarkable that drag-and-drop is found so rarely in programs that specialize in drawing. Thankfully, this seems to be changing, as more programs adopt this idiom.



There are several variations of drag-and-drop, and they are only a subset of the many forms of direct manipulation. The characteristics of drag-and-drop are fuzzy and difficult to define exactly. We might define it as “clicking on some object and moving it elsewhere,” although that is a pretty good description of repositioning, too. A more accurate description of drag-and-drop is “clicking on some object and moving it to imply a transformation.”

The Macintosh was the first successful system to offer drag-and-drop. A lot of expectations were raised with the Mac’s drag-and-drop that were never truly realized for two simple reasons:

1. Drag-and-drop wasn’t a system-wide facility, but rather an artifact of the Finder, a single program.
2. As a single-tasking computer, the concept of drag-and-drop between applications didn’t surface as an issue for many years.

To Apple’s credit, they described drag-and-drop in their first user interface standards guide. On the other side of the fence, Microsoft not only didn’t put drag-and-drop aids in their system, but it wasn’t described in their programmer documentation. Nor was it implemented in their Finder equivalent, the notoriously brain-dead MSDOS.EXE, the first Windows shell. The only drag-and-drop anywhere in Windows was in the simple paint utility distributed with the system. Yet again, Microsoft shipped an operating system—a standard-defining tool—but abdicated their responsibility for adequately defining collateral standards. I’m not ungrateful, as Windows was still by far the best thing around on the PC platform. Still, had Microsoft defined even some rudimentary standards, the drag-and-drop world would have evolved stronger and more rapidly.

It wasn’t until Windows 3.0 that any drag-and-drop outside of MSPAINT.EXE appeared. The new File Manager and Program Manager programs supported a rudimentary form of drag-and-drop. You could drag icons around in the Program Manager and files and directories around in the File Manager. Wonder of wonders, you could also drag an EXE file\* from the File Manager into the Program Manager and create an icon, although few users knew this. This disappointing lack of design leadership has resulted in an industry-wide sluggishness to embrace drag-and-drop, much to our software’s detriment.

After ten years, though, Windows is finally getting a drag-and-drop standard. It is not strictly a part of Windows, but rather a part of the OLE 2.0

specification. To get a community of third-party technology, there is something even better than having a solid set of library routines that enable them to build applications without having to invent the wheel. Libraries have ever been made available in the Windows world, but it is so large and frustratingly complex that there is no drag-and-drop standard will become either lost or badly implemented. This unfortunate bind will only be solved if a vendor encapsulates the functionality of drag-and-drop in an easy-to-program package, then makes it available to developers.

I find it amusing that the Microsoft style guide describes drag-and-drop. It makes it sound like a simple and commonly known task, describing how to put on your shoes in the most straightforward way possible.

## Dragging where?

Fundamentally, you can drag-and-drop something from inside your program, or you can drag-and-drop something from one program into some other program. I call these **interior drag-and-drop** and **exterior drag-and-drop**, respectively.

Interior drag-and-drop can be made pretty simple from a coding point of view. Exterior drag-and-drop requires more sophisticated support because both programs must understand the concepts, and they must be implemented in concert. We’ll look about the exterior variant after we get a look at interior drag-and-drop.

I classified repositioning as a direct-manipulation operation in the last chapter. Now we will discuss the remaining types of drag-and-drop. Primarily, there are two: master-and-target and master-and-slave.

## Master-and-target

When the user clicks on a discrete object and drags it to another object in order to perform a function, I call it **master-and-target** drag-and-drop.

The object within which the dragging originates is called the master object. It is the master object, which will be a window. If the master object is a window, the target object will be a window or a control within a window.

is the window. When the user ultimately releases the mouse button, whatever was dragged is dropped on some **target object**.

The main purpose of the term “master-and-target” is to differentiate this operation from the kind of drag-and-drop operations we find in drawing and painting programs, where tools and graphical objects are dragged around on an open canvas. Master-and-target is a more function-oriented idiom, where manipulating logical objects represents some behind-the-scenes processes. The most familiar form of master-and-target drag-and-drop is rearranging icons in the Program Manager or in the Macintosh Finder.

### Dragging data to functions

Instead of dragging a file or folder to another folder, you can drag it to a gizmo that represents a function. This idiom is arguably the most famous expression of direct manipulation because of the Macintosh’s familiar trashcan. Windows 95 copies this familiar idiom with its “recycle bin.” Someday, as we build software with better object-orientation, we’ll be able to drag-and-drop objects onto gizmos representing functions other than just delete. Imagine targets representing a cloner, an archiver, a file compressor, a faxer or a contents-indexer.

Notice that all of the idioms in the above paragraph involve exterior drag-and-drop, because the target objects are separate programs. Within a single program, the code knows what objects are draggable—usually one type—and any function gizmo that it gets dropped on will easily handle it. In an exterior drop, the master object can come from any program, and the target gizmo may well not have any direct knowledge of the originating program or the dropped object. The target must be able to handle the unknown object in some reasonable way without necessarily understanding what it is or what is in it. The Program Manager, for example, can do this because it knows that it will only be handed files. What would it do if it were handed a paragraph of dragged text from a word processor, for example? If it can’t handle the text, it isn’t truly exterior capable. To Microsoft’s credit, the Recycle Bin in Windows 95 can actually accept paragraphs of text dragged from Word or cells dragged from Excel. I have not yet been able to determine whether these are generic operations or just code specific to Microsoft applications.

To be truly exterior capable, an object must be able to accept a drop of anything from any other object, regardless of the originating program. At first, this

to be. Mostly, it’s a matter of defining interface. If an object is dragged to an object, all the target object has to do is “drop” to the master object. The two objects then negotiate the data because it is unreasonable to expect every object to understand every program’s proprietary formats. If the master object offers the data in its internal format. Another Microsoft product has to decipher this format, but a Brand X product might get object politely demurs—not to the drop, but to the data’s contents. Excel, the master, must then re-offer the data in generic formats: SYLK, CSV, ASCII. The target object must accept SYLK or CSV, but by convention, it must accept a common denominator format on all platforms. Every program must minimally accept ASCII, simple bitmaps, pointers to functions. Objects that hope to become successful must accept many more formats than that, but these are not negotiable with everything. Even an audio file, for example, is a simple pointer to a disk file. I call exterior drag-and-drop that supports this type of haggling over formats **negotiated**.

I call protocols like those in the Windows 3.x Program Manager, which don’t negotiate formats, **known**.

### Dragging functions to data

Proper, negotiated, exterior drag-and-drop capabilities include dropping functions onto data as well as dragging data onto functions. Defining the scope of such actions can be pretty concrete data, but it can still be generally quite abstract. You could click on the italic button on the toolbar and apply it to a spreadsheet. Clearly, the user’s intent in this action is to make that cell to italic. Part of the format negotiation involves recognizing a function as a valid drop value. Conceptually, there is a difference between the function “delete” and the function “italicize.” The program deletes its internal copy of the data and hands it back. In other, the target program hands a copy of the data back. This converts the text to italic and hands it back. This is how Excel’s window can be dragged onto text in Word. Word knows what to do with it. Or, more meaningfully, the text dragged onto the text in Microsoft Word. Once the



# 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.