JAMES FELICI
FOREWORD BY FRANK ROMANO

The Complete Manual *of* Typography

SECOND EDITION

A GUIDE TO SETTING PERFECT TYPE

"The ultimate typographic tool: a concise, beautiful book that pulls together everything you need to produce great typography."

FRANK ROMANO
ROCHESTER INSTITUTE OF TECHNOLOGY, SCHOOL OF PRINT MEDIA

"Dangerously good book on typography. 'Dangerous' because there is enough well-presented information in this volume to set you on the path to typography snobbery. This book is an excellent read and reference volume for any designer, print or web."

—NORA BROWN, Nora Brown Design

"Felici elegantly and painstakingly sets out to demonstrate how to set type 'perfectly' in a digital age. This is the book that answers all the questions you wanted to ask, but also demonstrates all the steps you need to pursue to achieve a kind of typographic perfection."

—MARGARET RICHARDSON, FontShop

"Buy this book, read it cover-to-cover, then keep it handy. You'll be surprised at what a difference it can make in the appearance of your work, both print and web."

—PETER BAUER, *Photoshop User*

"*The Complete Manual of Typography*, by James Felici, condenses timeless wisdom and timely technology into one complete guide. It explains everything about type designs and usage. If you had only one book on typography, this should be it."

—JAY NELSON, *Design Tools Monthly*

"Reading this book is like sitting down with a longtime typesetter and going over the details of a complex job. Most people will use it as a reference—which it is—but reading any section straight through is rewarding. The writing is clear and straightforward, and Felici has obviously thought long and hard about everything he deals with here."

—JOHN D. BERRY, CreativePro.com

"This excellent book discusses how type should look and how to set type like a professional."

—LINDA BUSHYAGER, *HiTech Review*

"What Felici's book does is show the importance to the reading experience of type that is well set on the page. It is copiously illustrated and elegant in design, and, I confess, I savored each of its 300 pages."

—DAN BARNETT, Musable Blog

"This is a superb reference book and should be often consulted by those who take pride in typography."

—PHILLIP PARR, Cider Press

"James Felici deserves a special place on every computer user's desk because with the power to put words on paper there comes a responsibility to do it well. For the ultimate guide to setting perfect type, you'll need *The Complete Manual of Typography*."

—FRED SHOWKER, DTG Magazine

"If nothing else, this book will make interesting reading for people who love to read books and think about the written word. For me, I wouldn't be without it, no matter the cost. This is one of my better reference books, and I Love Type."

—GEORGE ENGEL, Foxwood Estates Computer Club

"While Felici has abundant experience setting type in almost every format used in the twentieth century, he takes the capabilities and possibilities of the computer as a starting point for a very lucid and practical discussion of how to get the best possible type from software. The book contains one of the few really clear explanations of hyphenation and justification settings and how best to use them, as well as very practical and contemporary advice on issues such as line length and text color."

—FONTS ANON

"It covers all aspects of type design and applications of them in print and screen. This is like a master course in the finer points of typography. For a book that covers the historical tradition as well as digital innovations, this is a remarkable achievement."

—ROY JOHNSON, Mantex.co.uk

The Complete Manual *of* Typography, Second Edition

The Complete Manual *of*

SECOND EDITION

Typography

A GUIDE TO SETTING PERFECT TYPE

JAMES FELICI

The Complete Manual of Typography: A Guide to Setting Perfect Type, Second Edition
James Felici

This Adobe Press book is published by Peachpit.
For information on Adobe Press books, contact:

Peachpit
1249 Eighth Street
Berkeley, CA 94710
510/524-2178
510/524-2221 (fax)

For the latest on Adobe Press books, go to www.adobepress.com
Peachpit is a division of Pearson Education.
To report errors, please send a note to: errata@peachpit.com

Editor Rebecca Gulick
Production Editor and Compositor David Van Ness
Cover and Interior Designer Frances Baca with Mimi Heft
Copy Editor Karen Seriguchi
Proofreader Patricia Pane
Indexer Jack Lewis

This book is set in Monotype Perpetua and Linotype Syntax, both from Adobe Systems. Perpetua is a trademark of the Monotype Corporation registered in the U.S. Patent and Trademark Office and may be registered in certain other jurisdictions. Syntax is a registered trademark of Linotype-Hell AG and/or its subsidiaries.

ISBN-13: 978-0-321-77326-5
ISBN-10: 0-321-77326-8

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

for Jennifer

Foreword

Typography is what communication looks like.

But it is almost impossible to look and read at the same time because they are different perceptions. There is beauty in the language and beauty in the way it is presented. It all started about two millennia ago.

In AD 111, there was one typeface. It was the inscriptional seriffed lettering on the Trajan Column. In AD 2011, there are almost 200,000 fonts (most of them based on Garamond).

The Romans chiseled type into granite and made it monumental. Later, Jenson engraved type into metal and made it elemental. He went from columns of stone to columns of type.

There is a difference between type and typography. Typography was born because Gutenberg wanted to make his Bible appear handwritten. It was the first major publishing scam—pages reproduced as printed type at hand-calligraphy prices.

Typography is the use of type to advocate, communicate, celebrate, educate, elaborate, illuminate, and disseminate. Along the way, the words and pages become art. Type and typography fostered books, magazines, catalogues, newspapers, forms, and a plethora of promotional materials.

Type and typography—what you do and how you do it—are both science and art. There are rules, most of which are ignored. There are tools, most of which are unknown. But now you have the ultimate typographic tool: Jim Felici's knowledge at your fingertips.

As we edge toward 600 years of linear text and phase from paper to screen, the principles of good typography have not changed even as the technology of typography continues to change.

Type and typesetting went from metal to wood to film to dots. We set individual letters, lines of letters, and then pages of letters. We went from mechanical machine to mainframe to mini to desktop computer. We went from bitmaps to programmed curves and splines. We went from PostScript to TrueType to OpenType.

In only a few years we wiped out the entire typesetting industry, and typesetting became the province of the creative originator. The most demanding

type buyers became less demanding. We saw typewriter inch marks instead of real quotes and two hyphens substituting for em dashes. Forget about en dashes and real small caps and good H&J. Eventually, the industry did give us professional font sets, and programs automated many typographic processes.

There was a time when Courier, a monospaced typewriter face, was the most used typeface on the planet. Today that distinction belongs to a combination of Times and Helvetica. The most used faces are still the classics.

The letter and the numeral and the symbol begat the glyph, and the number of glyphs in a font multiplied—real small caps, old-style figures, gobs of diacriticals, and dingbats galore.

Those who work with type have to catch up with both what is old and what is new. Fortunately, you have the solution in your hands: a concise, beautiful book that puts together in one place everything you need to produce great typography. Thanks, Jim.

—FRANK ROMANO
Professor Emeritus
Rochester Institute of Technology
School of Print Media

Contents at a Glance

This page intentionally left blank

Table of Contents

Introduction

This book is about how type should look and how to make it look that way. It primarily covers type in print, which is where the art of typography reaches its highest form of expression. But people who read on computers, e-book readers, or any other electronic device need all the typographic help they can get, so setting type for screen display also gets its due.

The book is organized so that you can approach it in two ways: as a textbook to read from cover to cover, or as a reference guide to jump into at any point as need dictates. It has a wonderful index.

This is not a style guide, but an execution guide. It doesn't explain why you might choose to use the typeface Bembo over Garamond, but rather, having made that choice, how you can set Bembo in the best possible way.

The rules of typography are centuries old, and although the technologies have changed, the goal has always remained the same: a beautiful setting in the service of a pleasant and fruitful reading experience. So while this book explains in very practical terms how to use today's computerized tools, I've written it to outlast them. It's been completely updated since the first edition appeared in 2003, and as in the original, references to specific programs have been kept to a minimum (although the capabilities of all the major programs have been taken into account). Programs change, but the lessons in this book will be just as applicable to version 20.0 of your software as they are to the version you use today.

Not all of the capabilities discussed in this book exist in every program or system. But none of them are fantasies—they all exist somewhere. Every typographer and typesetter has to hope that they all will converge in one program as soon as possible. In the meantime, I've included scores of workarounds to wring good type out of uncooperative programs.

Beautiful type comes from attention to myriad tiny details. It's built up a fraction of an em at a time, through hundreds of decisions whose geometry belies their gravity. It requires, as a colleague once wrote, a heart hardened against accusations of being too fussy.

—JAMES FELICI

This page intentionally left blank

CHAPTER 4 About Fonts

Typefaces are what you get to admire after your work is finished, but fonts are the tools you have to wrestle with in the meantime to get the job done. Computer operating systems and applications have made it much easier to work with fonts, but the process is still quite technical. Working with fonts forces you to learn more about your computer than you probably want, but everything you need to know is in this chapter.

The Two Basic Kinds of Fonts: Outline and Bitmapped

Digital devices—computer monitor screens, desktop printers, imagesetters—create images out of dots. The simplest way to create type for one of these devices is to draw a picture of every character as an array of dots and store these drawings in a font. Then all a device has to do to image the type is to copy those dots into place on the screen or page. When this technology was first figured out, each one of those dots was represented by one *bit* of computer data—a simple yes/no choice of whether to image a dot or not. Images created from these predrawn, prearranged arrays of dots were called *bitmaps,* and fonts using this trick were called *bitmapped* fonts.

FIGURE 4.1 In a bitmapped font, each dot in the drawing of a character is mapped against one pixel of the output device. As the device's resolution increases, the dots of the bitmapped font become smaller and smaller, and the character follows suit. In this illustration, a character bitmapped for the screen appears at left. If this bitmap is imaged on a desktop printer, it shrinks to the size in the center. At a typical imagesetter resolution, it shrinks to the size on the right.

Bitmaps are a clever and simple approach, but the more dots a bitmap contains, the more computer data it requires. As the resolution of the device increases or the size of the character images increases, the number of dots grows geometrically: Doubling the size of a character quadruples the number of dots. You also need a separate set of bitmaps—a separate font—for every size of type you want to create. And the bitmaps designed for one resolution will appear much smaller when imaged on a device with a higher resolution, where the dots are much smaller (see Figure 4.1). To image a single typeface at the same range of sizes on a computer screen, a desktop printer, and an imagesetter, then, would require hundreds of bitmapped fonts.

The solution is to store the descriptions of the characters as a set of outline drawings. *Outline fonts,* which do just this, store character images as outlines described mathematically as a series of curves and straight-line segments. (These line segments are sometimes called *vectors;* and the fonts based on them, *vector fonts*.) These outlines can be mathematically scaled to any size without distorting the shapes or proportions of the characters. The scaled outlines are then colored in with dots of the size created by the device that the type is being imaged on: around 100 dots per inch (dpi) for a computer screen, approximately 600 dpi for a desktop printer, and well over 1,000 dpi for imagesetters.

While it was once common for operating systems to use bitmapped fonts for screen display, they now generate screen type from the same outline fonts used for high-resolution printing. Some fonts may contain sets of hand-tuned bitmapped screen fonts for use at small sizes because they're more legible than those generated by your computer. But these embedded screen fonts are not apparent to the user, and you don't have to concern yourself with them.

What's in a Font?

A font contains all the information needed to position and image the characters that it represents. How a computer operating system and an application program team up to use this information is covered in detail in Chapter 7. Here we're just concerned with what's inside a font and what it means to you as you set type.

The most important constituents of a font are the character outlines themselves. The entire collection of characters in a font is called its *character set*. For most *alphanumeric* fonts (that is, the ones used for text containing letters and numerals), character sets are standardized to a degree. Nearly all of these fonts share a basic set of characters, although they may contain optional extra characters as well. Figure 4.2 shows the core character set of a standard text font as well as some common variants used by various font vendors. Fonts based on Unicode (see the section on OpenType fonts on page 55) may contain additional characters beyond these basic collections.

The character outlines in a font are size independent. Inside each font a *width table* lists the horizontal space allotted to each character, as measured in fractions of an em. Computer programs use these widths to calculate how to fill lines with type, adding up the cumulative widths of the characters on a line until the line is filled.

A font may also contain tables for the widths of other members in its family. This is typically the case for the "regular," or roman text-weight, member of a family. These tables enable a computer program to compose type for all four members of a family—regular, italic, bold, and bold italic—using only the regular font. The computer's operating system, using the widths of the other family members, can synthesize false italics, bolds, and bold italics for onscreen display, relying on width tables in the regular font for getting the spacing and positioning right. The typesetting program, which relies only on the character widths, follows suit and can make appropriate decisions about how much text will fit on a line and how lines should be broken. When it comes time to print, all the necessary fonts will have to be present, as their outlines will be needed to image the type (see Figure 4.3). But to simply compose the type onscreen, only the regular-weight font is needed. The relationship between application and operating system is detailed in Chapter 7.

A font also contains a *kerning table,* which lists specific letter pairs and how the typesetting program should adjust the spacing between them. Kerning adjustments are also expressed in fractions of an em, which enables them to function at any point size. For more information about kerning, see Chapter 11.

PostScript Type 1 character set

a b c d e f g h i j k l m
n o p q r s t u v w x y z
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 0 ? ¿ @
! ¡ # $ ¢ ¥ £ € % ‰ ^ & *
+ ± ÷ = - – — ( ) [ ] { }
_ ; : ' " \ | , . < > / `
~ ™ *f* ¶ • ª º ® † “ ” ‘ ’
ß © · ° ¬ … ‹ › « » ˇ ⁄ ¤
fi fl ‡ ° · ‚ „ æ Æ œ Œ ı §
ˌ ˛ - ` ´ ¨ ˆ ˇ ˜ ″ ç ø å
à è ì ò ù á é í ó ú ä ë ï
ö ü ÿ â ê î ô û ã ñ õ Ç Ø
À È Ì Ò Ù Á É Í Ó Ú Ä Ë Ï
Ö Ü Ÿ Â Ê Î Ô Û Å Ã Ñ Õ Ð
ð Þ þ Š š Ý ý Ž ž ¼ ½ ¾ ¹
² ³ ¦ - × µ

characters added in Adobe Standard OpenType character set

Ł ł Ω π ∏ Δ ∂ Σ √ ∫ ∞ ≠ ≈
≤ ≥ € ℮ ℓ

characters added in Bitstream OpenType character set

Č č Ğ ğ İ Ş ş ₣

characters not included in Monotype Basic OpenType character set

- ` ´ ° · ¨ ˛ ‚ ˆ ˇ ˜ ″ ˘
∏ Σ

FIGURE 4.2 At the top is the standard character set of a PostScript Type 1 font used by most vendors. Although such a font can nominally contain 256 characters, 33 "slots" in the font are taken up by commands such as backspace and delete, and 2 by the word space and nonbreaking space. Below it are the additions made to create the standard character sets for OpenType fonts from Adobe and Bitstream. Monotype uses the same character set as Adobe for its Basic OpenType fonts, with the exception of the characters noted at the bottom.

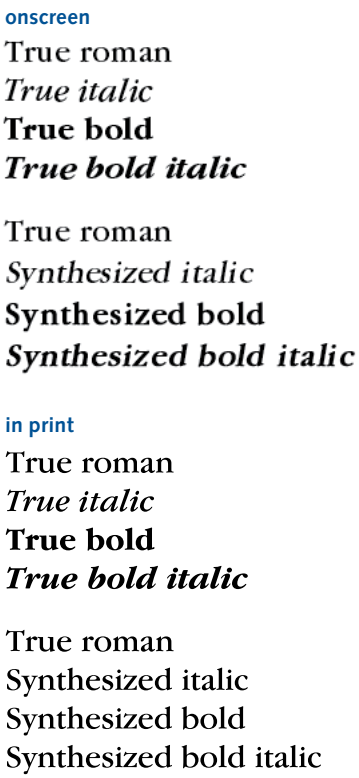

FIGURE 4.3 In this illustration, the top four lines of screen type were generated from their actual fonts. The computer generated the second set of four lines by interpolating the outlines of the plain roman font. You can see that the "italics" are simply obliqued roman characters.

The high-resolution lines at the bottom show what you get if you try to print the two samples. With all the fonts available, printing proceeds normally. But without the outlines for the other three members of the family, the printer uses the plain roman font for all four lines.

Font Formats

Ultimately, what's inside a font depends on its format. The word *format* has two meanings in computer type. First, it can refer to the platform for which the font was designed. For example, two fonts with the same data for the same typeface may have different file formats depending on whether they're designed for use on an Apple Macintosh or a Windows PC. Until the development of the OpenType font format, fonts were created to meet the data-structuring needs of one platform or the other, and a font designed for one machine would not work on the other. A single OpenType font file will work on either a Mac or a PC.

Another kind of font format reflects how the typographic information itself is described and organized. The three leading font formats today are PostScript, TrueType, and OpenType.

POSTSCRIPT FONTS

PostScript fonts are written in the PostScript page description language, and they need to be processed by a PostScript interpreter before they can be imaged. (See "The PostScript Model" in Chapter 1 for more information on PostScript interpreters.) For high-resolution printers and imagesetters, this interpreter is generally built into the device itself; it's a separate onboard computer dedicated to turning PostScript code into printable output. For lower-resolution devices, such as computer monitors and desktop printers, PostScript fonts can be imaged by a PostScript interpreter built into the operating system. PostScript fonts are generally accompanied by a set of bitmapped fonts for screen display, and unless these screen fonts are installed alongside the outline fonts, your computer cannot image their type. Even though your computer may not use the screen fonts' bitmapped images, it relies on the font metrics contained within the screen fonts to compose type using their companion outline fonts. This is an artifact of older technology, but it continues to function perfectly well.

The several kinds of PostScript fonts are distinguished from one another by number. The only one you're likely to come across is *Type 1,* and it's only mentioned here because of references you may come across to "PostScript Type 1" fonts. In publishing and typesetting contexts, when you talk about a PostScript font, it's assumed you're talking about the Type 1 variety.

Until the advent of the OpenType font format, PostScript fonts were the standard of the publishing industry. Today the PostScript format has been completely overtaken by OpenType, and most type vendors, including Adobe, have converted their entire libraries of PostScript fonts into the OpenType format. PostScript fonts continue to be fully supported by applications and operating systems, which is a good thing, because there are literally millions of them still

in circulation and daily use. They are, however, platform specific, and different versions of a font are required for Macintosh and Windows.

TRUETYPE FONTS

For a few years in the late 1980s, the typesetting world had in PostScript a single, standard *font format* for the first time in its history. It wasn't to last. For a combination of primarily commercial but also technological reasons, Apple Computer and Microsoft collaborated to create a new font format: TrueType. The new format enabled both companies to build outline font-imaging capabilities into their respective operating systems without being beholden to Adobe.

TrueType introduced many improvements over the PostScript format. The most prominently touted was its *hinting,* instructions added to the font that tell the character outlines how to reshape themselves at low and medium resolutions in order to create character images of maximum clarity. (For more on hinting, see "Imaging PostScript Fonts" in Chapter 1.) Because of the high quality of these hints, TrueType fonts were and still are typically delivered without any hand-drawn, bitmapped screen fonts. Screen type generated from the font's character outlines is generally quite legible even in small point sizes.

TrueType also allowed for larger character sets. The PostScript font format had used a numbering system to identify the characters in its fonts based on a single byte of computer data, yielding a maximum of 256 distinct ID numbers. (Fonts of this kind are still referred to as *single-byte fonts.*) TrueType introduced a two-byte numbering system, which allowed much larger character sets by creating over 65,000 unique ID numbers.

This made plenty of room for alternate forms of characters as well as allowing languages that rely on huge character sets (such as Chinese, Japanese, and Korean) to be supported by a single font. TrueType fonts are still included as a part of major operating systems, but most independent digital font foundries have shifted to OpenType because it allows a single font file to work under multiple operating systems. TrueType fonts are still platform specific, and a TrueType font created for use on a Mac will not work on a Windows PC, and vice versa. TrueType fonts use a different technology than PostScript fonts do for describing the outline shapes of characters, but any system that can image type from PostScript fonts can also image type from TrueType fonts.

MACINTOSH DFONTS

Many Macintosh-specific fonts use a file structure that predates OS X. In this structure, the file contents are divided into two parts: a *data fork* and a *resource fork.* Older versions of the Mac OS used data in the resource fork to tell (among other things) what application created a specific file. Mac OS X does

this by reading a file's filename extension, such as .doc. Dfonts are a variety of TrueType font that have no resource fork, and they are included in OS X for the sake of font compatibility with other computers running the UNIX operating system. (OS X, like Microsoft Windows, is based on UNIX.)

You can use dfonts just as you would any other Macintosh TrueType font. Documents formatted with them will not, however, display correctly on Macs running operating systems that predate OS X.

OPENTYPE FONTS

OpenType is a hybrid font format created by Adobe and Microsoft. It reconciles the differences in the PostScript and TrueType formats, allowing them to exist together in a single file. OpenType fonts are also written in a file format that allows the same font file to be used on either a Macintosh or a Windows PC. Crudely put, an OpenType font is a TrueType font with a "pocket" for PostScript data. An OpenType font can contain TrueType font data, PostScript font data, or (theoretically) both. Thus it has the potential to combine the best of both formats in a transparent way. The operating system of your computer will sort out the data in an OpenType font and use what's appropriate for it. A problem with OpenType fonts, as with the TrueType fonts that preceded them, is that from the outside there's no way to know what's inside. The original generation of PostScript fonts generally contained a standard character set with standard features. The TrueType format and, to an even greater extent, the OpenType format offer a wide range of optional features that may or may not be built into every font, although the core character set used in the original PostScript fonts has generally been retained. An OpenType font can contain anywhere from a handful of characters to more than 65,000. There's no way of knowing what a particular font contains or what it can do unless the features of the font are documented in some way.

OpenType fonts also enable a variety of so-called *layout features,* which give a typesetting program the ability to automatically substitute one character for another. Using an appropriate OpenType font, for example, a program can automatically convert the keystroke sequence 1/2 into a proper fraction: ½. Layout features are discussed in detail on pages 62–64.

WEB FONTS

The term *web font* does not refer to a specific font format but to fonts that have been extensively hinted for optimum legibility when displayed on computer screens and other electronic devices. Some have been designed from scratch for electronic display, while others have been adapted retroactively.

Popular web standards permit designers to specify the use of particular fonts when their pages are displayed, even though these fonts are not embedded in the file or necessarily available on the device displaying it. In this sense, web fonts are also understood to be those that exist on web servers for real-time use for imaging online documents that call for them. Some of these are available for free, but others are available only under license, with a fee paid for their use; they are, in effect, rented.

Web fonts are also discussed in Chapter 17, in the context of the Cascading Style Sheet standard used to structure many web documents.

Unicode: The Underlying Technology

All computer programs identify characters by number. International standards correlate every number to a unique character, so that a computer file from Europe, for example, can be properly typeset in Asia. It took decades before a single standard international numbering system was established: Unicode. Both TrueType and OpenType fonts use Unicode numbers to identify their constituent characters.

The goal of Unicode is to assign a unique ID number to every character, linguistic symbol, or ideogram in all of the world's languages, living or dead. The number of such IDs now exceeds 100,000.

To facilitate backward compatibility, and to support legacy documents, today's computing systems still suffer from vestiges of earlier numbering systems. The first of these was ASCII (the American Standard for Computer Information Interchange), which used the numbers 0 through 127, as shown in Figure 4.4. The original desktop computing systems—including Microsoft DOS and Windows and the Apple Macintosh OS—used one-byte numbering systems that were consistent through the ASCII range but differed in the ID numbers assigned to the other 128 characters a font could contain. This made communications between the two platforms needlessly complicated, with characters often incorrectly displayed on a nonnative system.

For technical reasons, the ID numbers assigned by Unicode are written in *hexadecimal* format. Hexadecimal, in addition to using the numerals 0 through 9 to express numbers, also uses the letters A through F. This allows 16 values to be expressed with a single character, like so: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. The letters following 9 represent 10, 11, 12, 13, 14, and 15, respectively, in our everyday counting system. In hexadecimal, the value expressed as 0010 (Unicode values are always expressed using four "digits") is the equivalent of 16 in our normal base-10 system.

Fortunately, you don't need to know anything more than this about hexadecimal notation, and even the preceding paragraph is added only to explain

32	(space)	**64**	@	**96**	`
33	!	**65**	A	**97**	a
34	"	**66**	B	**98**	b
35	#	**67**	C	**99**	c
36	$	**68**	D	**100**	d
37	%	**69**	E	**101**	e
38	&	**70**	F	**102**	f
39	'	**71**	G	**103**	g
40	(	**72**	H	**104**	h
41	)	**73**	I	**105**	i
42	*	**74**	J	**106**	j
43	+	**75**	K	**107**	k
44	,	**76**	L	**108**	l
45	-	**77**	M	**109**	m
46	.	**78**	N	**110**	n
47	/	**79**	O	**111**	o
48	0	**80**	P	**112**	p
49	1	**81**	Q	**113**	q
50	2	**82**	R	**114**	r
51	3	**83**	S	**115**	s
52	4	**84**	T	**116**	t
53	5	**85**	U	**117**	u
54	6	**86**	V	**118**	v
55	7	**87**	W	**119**	w
56	8	**88**	X	**120**	x
57	9	**89**	Y	**121**	y
58	:	**90**	Z	**122**	z
59	;	**91**	[	**123**	{
60	<	**92**	\	**124**	|
61	=	**93**	]	**125**	}
62	>	**94**	^	**126**	~
63	?	**95**	_	**127**	(delete)

FIGURE 4.4 Computers identify characters by numbers, and all systems agree on the meanings of 0 through 127, the so-called ASCII character set. The numbers 0 through 31, not shown here, are either unassigned or assigned to nonprinting commands such as return and backspace. The ASCII character set is printed on most English-language computer keyboards.

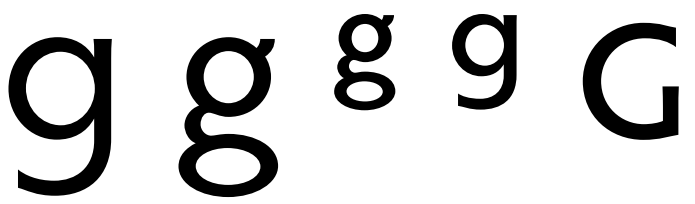

FIGURE 4.5 A single character with a single Unicode ID number can have several forms, each represented by a unique *glyph*. Here, a lowercase *g*—Unicode number 0067—from the typeface Hypatia Sans Pro can be represented by any of five alternate glyphs.

why Unicode character numbers look so peculiar when seen in a font browsing window.

Both Windows and the Mac OS now support Unicode as well as continuing to support the numbering schemes used in older font formats. This happens more or less transparently, although how you access certain characters in certain fonts will vary according to their format. This is described in detail later in the chapter, in the section "Finding the Characters You Need."

CHARACTER VS. GLYPH

An important aspect of Unicode is that it recognizes that a single character may have several forms, each one of which is represented by a distinct *glyph,* as shown in Figure 4.5. Unicode's main concern is clear communication, not typography per se, so it does not distinguish between a simple roman *A* and a decorated *A* used for design purposes. For Unicode, the goal is simply to accurately depict a capital *A* as a capital *A*. All capital *A*s, then, have the same Unicode number—0041—although they may be represented by alternate *glyphs.* Tracking which glyph you've chosen to use is the job of your typesetting or page layout application.

For this reason, computer tools used for browsing the contents of fonts are often called *glyph palettes,* and a given font's glyph set can be far larger than its character set.

Cross-Platform Font-Compatibility Issues

The legacy left by evolving font standards continues to bedevil the movement of document files between different computer systems. The only way to be sure that a typeset document appears on one platform exactly as it was designed on another platform is to create it using the same OpenType fonts from the same vendor on both platforms.

Font-Encoding Issues

How numbers are assigned to the characters within a typeface is referred to as a font's *encoding.* Before they supported Unicode, the Macintosh and Windows operating systems used different encoding schemes.

Not only did the pre-Unicode operating systems use different character-numbering schemes, but they also used different subsets of the basic Latin 1 character collection as their standard character sets. The Macintosh set and encoding scheme were called MacRoman; the Windows character set and encoding scheme were called Win ANSI. Although a vendor might sell identical fonts for both platforms, the Mac would allow its users to access one group of characters within a font, and Windows another. Figure 4.6 shows the characters that were unique to each platform.

Today's operating systems on both platforms allow access to all of these characters. But both the Mac and PC lack keystroke combinations that allow you to easily type their formerly inaccessible characters. For the sake of backward compatibility, and in respect for people's keyboarding habits, both operating systems act as if their old encoding schemes were still in use. To get access to the Unicode characters, you have to use special techniques (discussed in the next section).

Although Unicode is not a font encoding per se, it does provide applications on any platform with a standard way of indicating which characters to use. To assure the accurate representation of text as it travels through other computer systems, using Unicode-based fonts is a must.

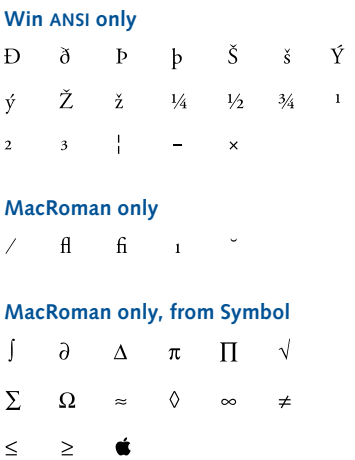

FIGURE 4.6 Of the basic PostScript Type 1 character set shown in Figure 4.2, only Windows programs have direct keyboard access to the group of characters shown at the top here. Only Macintosh programs can use the keyboard to access the ones in the middle group. The bottom group includes characters in the basic MacRoman encoding that appear to be in every Mac font, but they are actually borrowed from the Symbol font.

FOOTNOTE: THE MAC'S "BORROWED CHARACTERS"

When you're working with PostScript fonts (and many TrueType fonts) on a Macintosh, the MacRoman encoding borrows certain characters from the Symbol font (see Figure 4.6). Such characters seem to be a part of every font you use. The keystroke combination Option-D, for example, always yields a lowercase Greek delta: ∂. But the numbers assigned to these characters in the MacRoman encoding scheme point to blank "slots" in a Mac font. Calls for these numbers are diverted by the operating system to the Symbol font. That explains why these characters never match the style of the typeface you're working in (unless it happens to be Times Roman, upon whose design the seriffed Symbol characters are based).

This curious situation is unique to the Mac and unique to this small handful of characters. It's been largely corrected in most OpenType equivalents of older PostScript fonts through the incorporation of these formerly borrowed characters into their expanded character sets. The Mac OS now explicitly shows that it's using the Symbol font when you use the original keyboard commands to set these characters.

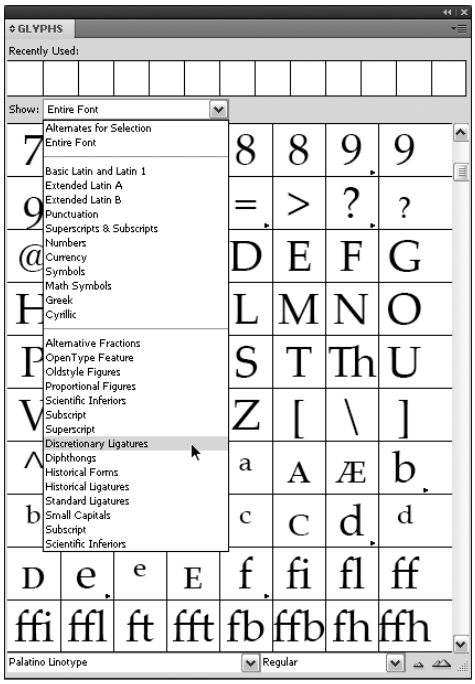

FIGURE 4.7 Adobe InDesign has its own glyph browser built in, and a pull-down menu lets you isolate classes of characters for faster searching amid extended character sets. These categories are specified inside the font itself, in this case TrueType Palatino from Linotype.

Finding the Characters You Need

Windows and the Mac OS offer tools to see exactly which characters are in a particular font. Applications themselves are beginning to offer the same services (see Figure 4.7). These font-browsing tools are indispensable in the absence of standard character sets and are the only way to get many of a font's glyphs into your documents.

Using Windows' Character Map

Windows' Character Map (found in the Programs > Accessories > System Tools menu) shows all the characters in a font in the form of a scrollable grid. From here you can select and copy a character or group of characters into your document. Selecting a character also indicates if there is a keystroke sequence you can use to access the character directly from the keyboard.

There are two kinds of such keystroke sequences. The first is based on the old Win ANSI font encoding used in older versions of Windows. These involve holding down the Alt key while you type the Win ANSI ID number of the character you want. For example, Alt-0233 yields a lowercase *e* with an acute accent: é. When you release the Alt key, the character will appear in your text.

Some applications—notably those from the most recent versions of the Windows Microsoft Office suite—use a similar technique based on the character's

Unicode number. In this case, you type the four-character Unicode ID for a character followed by Alt-x. This causes the program to reinterpret the previous four keystrokes and substitute the correct character. The keystroke sequence 00BD Alt-x, for example, creates the fraction ½. Note that the alphabetic characters in the Unicode ID can be typed in either upper or lower case.

Using typed commands to set specific characters is much faster than copying them out of the Character Map display, and making a written list of the character IDs you commonly use will save you a lot of time. Because there can be so many characters in a TrueType or OpenType font, the Character Map gives you the option to display only certain groups of characters at a time: those used for particular languages, for example, or numeric characters, including fractions and the characters for building them.

You can also use the Character Map's Search tool by using plain-English descriptions of the character you're after, such as *dash, fraction,* or *bullet*.

Using the Macintosh's Keyboard Viewer

The Macintosh's Keyboard Viewer displays a keyboard to show what characters are assigned to which keys. By default, this utility is buried away in the operating system. To make it easily available, open Language & Text within System Preferences (located in the Apple menu). Click the Input Sources tab, and in the pane, select the checkboxes next to Keyboard & Character Viewer and (near the bottom) "Show Input menu in menu bar." This causes a new icon (which looks like a flag, indicating your operating language) to appear on the right-hand side of the menu bar. The Input menu will give you access to the Keyboard Viewer utility.

With Keyboard Viewer open, holding down the Shift key changes the display to indicate which characters are available from each key with the Shift key held down. The same happens when you hold down the Option key or the Option and Shift keys simultaneously. With the four options—no Shift or Option, Shift, Option, and Option-Shift—the Mac OS enables each alphanumeric key to access four characters. As with Windows' Character Map, any characters you select in Keyboard Viewer can be copied into your documents.

The Mac OS also uses more complex keystroke combinations to access accented characters. In this process you hold down the Option key while you press a key that represents the accent you want to use: acute, grave, dieresis (umlaut), tilde, or circumflex. Then, in a separate action, you press the key of the letter over which you want to place the accent. At this point the accented character appears onscreen. To see where these accent characters are located, hold down the Option key with the Keyboard Viewer window open. You'll see that five keys appear with highlights. These are the accent keys.

In the Input Sources pane of the Language & Text System Preferences pane, you can choose which languages or keyboard layouts you'd like to include as alternate choices in the Input menu. If you add Russian, for example, and select it from the Input menu, Keyboard Viewer will show a Cyrillic keyboard layout. The same is true for French, British, or any other language- or nationality-specific keyboard layout.

THE MAC OS AND UNICODE

For Unicode fonts with large character sets, Mac OS has two principal tools: Font Book and Character Viewer. Font Book (found in the Applications folder) is a font manager, which you can use to control which fonts on your Mac are active at any time. Only active fonts appear in your programs' Font menus. By selecting Repertoire from Font Book's Preview menu, you can see the entire character set of any font currently installed on your Mac, a list of which appears on the left. You can't use Font Book to add characters to your documents, just to browse the contents of fonts.

Adding specific characters to documents is the job of Character Viewer. Character Viewer displays all of the characters in all of the fonts installed on your system. To find a particular character, you can browse by category or use the Search field. Character Viewer will show samples of your target character in all the installed fonts on your system. Double-clicking on the one you want inserts it into your text.

You can also use the Unicode ID number of a character to access it directly from your keyboard. To do this, once again open the Language & Text pane from within System Preferences and click the Input Sources tab. In the selection list put a check mark next to Unicode Hex Input, which adds this option to the Input menu. With Unicode Hex Input selected in the Input menu, holding down the Option key and typing a character's Unicode ID adds that character to your document. The downside of Hex Input is that when this input option is activated, you lose the ability to use the familiar Option and Shift-Option keyboard character-access commands. Nevertheless, if you maintain a list of commonly used Unicode numbers for hard-to-access characters, switching options in the Input menu is faster than finding the characters using a glyph palette.

Application Glyph Palettes

Page layout programs offer their own tools for browsing the contents of fonts and copying selected characters into document text. These are similar in concept to Windows' Character Map, but they offer easier access to alternate

glyphs where they exist for certain characters. You can usually create collections or sets of commonly used characters for which you have no direct keyboard access.

"Expert Sets" and Alternate Fonts

Before Unicode and OpenType came along, most fonts were restricted to 256 characters, and Latin-based alphanumeric fonts all contained a standard character set. (TrueType fonts have long been able to have larger character sets, but most font vendors—for the sake of compatibility—matched the character sets of their TrueType fonts to the sets of the PostScript fonts.) Those restrictions, though, were without historical precedent, and some typefaces had many characters for which there was no room in the standard font layouts.

The solution to the problem was to create companion fonts for certain typefaces, fonts that contained alternate characters. These companion fonts are called *expert sets* or *alternate fonts*. They include such characters as old-style numerals (which have varying heights, and some of which have descenders) and small capitals (scaled-down versions of capital letters made to be used amid lowercase type, where they are less obtrusive than full-size capitals). Other common expert-set or alternate characters include ligatures (tied letter combinations) and swash characters (with exaggerated terminals). Examples are shown in Figure 4.8.

The layouts of these fonts are not standard, so they're usually sold with a chart showing which keystrokes yield which characters. Unfortunately, Unicode-based character-locating utilities won't help, since most of these fonts predate Unicode, and many of the characters do not have standard Unicode numbers assigned to them in any case.

Expert-set fonts are also troublesome to use because they require a change of font, often for a single character. Macro programs or utilities—which enable you to program a key or screen button to execute a series of commands—are indispensable for dealing with expert-set fonts, as well as with pi fonts, for which you also need two changes of typeface to set a single character.

Fortunately, most typefaces with such extended character sets have been re-released in OpenType format, with their alternate sorts now rolled into a single font.

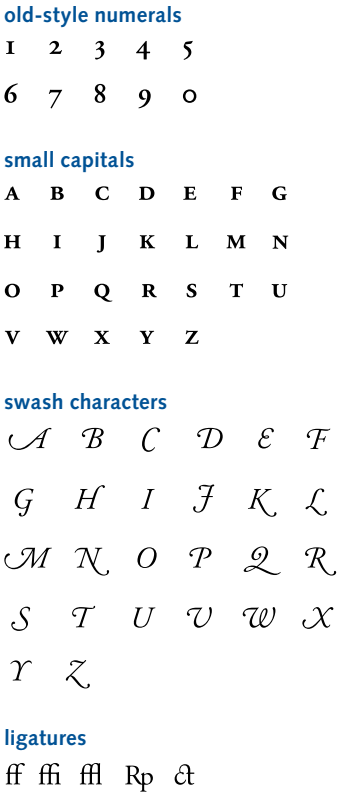

FIGURE 4.8 Expert-set fonts and alternate fonts contain characters that belong to a typeface but for which there's no room in the face's principal font. The characters shown here are drawn from the alternate and expert-set fonts for Adobe Garamond. With the larger character sets made possible by the TrueType and OpenType font formats, alternate fonts will eventually fade away, and these kinds of characters will be united with the font's standard characters in a single font.

Characters outside the Unicode Standard

The encoding issue brings up a murky side of Unicode: namely, if Unicode assigns specific numbers to specific characters, what happens when a type designer creates characters that aren't accommodated in the Unicode list?

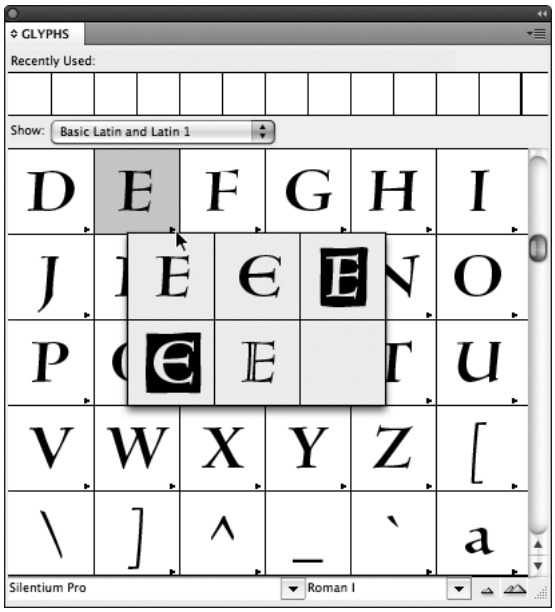

FIGURE 4.9 One Unicode number can point to several alternate versions of a single character. Here, four alternate forms of the *E* have been added to Adobe's Silentium Pro OpenType font, and they pop up from the basic character display grid in InDesign's character browser.

Such characters include many of those normally found in expert-set and alternate fonts.

The answer is that the Unicode scheme contains a range of numbers designated for "Private Use," and here a type designer can add customized characters. The meaning of these Unicode numbers, then, will vary from font to font. To simplify the arrangement, OpenType fonts can also contain links among characters to make it clear to an application or operating system that a particular character is actually an alternate form of one of the characters in the standard Unicode encoding. Figure 4.9 shows how an application can offer the choice of alternate characters to the user. By adding several planes to each character slot, the size of the total onscreen character grid is cut down to more manageable proportions, and characters can be found in logical places.

Look forward to the day when keyboards have illuminated readouts on the keys, so that when fonts change, keyboard layouts will change and the new character assignments will appear on the keys themselves. With the advent of large-character-set Unicode fonts, no one will be able to be a touch typist all the time anymore.

OpenType Layout Features

The OpenType font format was designed with internal structures that allow a type designer to populate a font with alternate forms for certain characters. These *layout features* allow you to have OpenType-savvy programs automatically substitute alternate characters when they're available. These substitutions can

be restricted to a selected passage of text or applied globally to entire documents. The more common layout features are listed here. The uses of such alternate glyphs are discussed in detail in Chapter 13.

SMALL CAPS

Reduced-size capital letters are used for setting certain kinds of type, including acronyms (NATO) and certain abbreviations (A.M., P.M.). Capital letters that have been electronically scaled down in size are ill proportioned in this role, so specifically designed small capitals should be used when available. With the OpenType small caps option turned on, any capital letters in the text will be converted to small caps, so this control should be applied locally only, to specific capitals that need to be set in reduced size.

ALTERNATE NUMERALS

The standard numerals contained in most typefaces all sit on the baseline and all have the same width. These are called *tabular lining numerals*. Alternate forms with proportional, varying widths—*proportional lining numerals*—may also be available in a font. Other alternate forms include old-style (or lowercase) numerals, in both tabular and proportional styles. Leaving the OpenType option set for Default Figure Style will set numerals in the preferred style designated by the typeface designer.

AUTOMATIC FRACTIONS

When this option is enabled, OpenType fonts that contain the necessary numerator and denominator glyphs can be used to automatically generate fractions. The keystroke sequence 12/25, for example, will be converted to 12⁄25.

ALTERNATE LIGATURES

Ligatures are fused characters designed to alleviate certain awkward character-shape interactions and sometimes used for historical or linguistic reasons. All text fonts include the common *fl* and *fi* ligatures, but some fonts contain many more. These can be made to appear in the text by turning on this layout feature. In most cases, this will be a global selection, affecting the entire document.

SWASH CHARACTERS

Swashes are exaggerated extensions to the strokes of certain characters, both upper- and lowercase. Selecting this option substitutes them for their normal undecorated forms.

SUPERSCRIPTS AND SUBSCRIPTS, ORDINALS AND SUPERIORS

Superscripts and subscripts are reduced-size letters and numerals used in mathematical and scientific notation, such as $E = MC^2$ and H_2O. Ordinals are alphabetic characters used for indicating numeric values such as 1st (in English) and 1º (in Spanish). How superscripts, subscripts, and ordinals align relative to each other and to full-size text varies from typeface to typeface.

TITLING AND CASE-SPECIFIC FORMS

Titling characters are designed for use in large display sizes. Sometimes they are available only in uppercase forms. Case-specific alternates include characters such as elevated hyphens for use in all-caps material.

CONTEXTUAL ALTERNATES AND POSITIONAL FORMS

In some settings, principally non-English and particularly Arabic text, the shape and alignment of a character vary according to where in a word it appears. In such cases, contextual alternate forms are used.

SLASHED ZERO

When available, a slashed zero (Ø) can be substituted for a normal zero to avoid confusion with a capital O.

STYLISTIC SETS

Some alternate glyph categories are essentially "none of the above." These sets are designated by the typeface designer and may consist of a single glyph, such as the historical medial *s*: ſ. They may also include sets of lowercase characters with longer or shorter ascenders and descenders. Their contents can be viewed from within application glyph palettes.

Identifying Font Formats

If you've read this chapter from the beginning, you'll realize that not all fonts behave the same way. PostScript fonts, TrueType fonts, and OpenType fonts all have their own idiosyncrasies, not to mention some major functional differences. It's important to be able to tell them apart.

If you're looking in the folders where the operating system stores them, you can distinguish among the three formats relatively easily. Some applications display icons alongside the entries in their Font menus, but such displays are

not standardized and not always perfectly clear. In general, it's better to know the formats of the fonts you use before you install them in your system and to create a method for keeping track of what's what. Fonts in different formats may appear with identical names in your Font menus, and having two such fonts listed side by side is something you want to avoid. Furthermore, it's entirely possible for an operating system to fail to distinguish between two fonts of the same name in different formats and to list just one of them in an application's Font menu. Not only won't you know that there are two fonts with the same name on your system, but you also won't know which one you're getting.

At one time, only OpenType fonts containing PostScript font data (so-called PostScript-flavored OpenType fonts) ended with the filename extension .otf. This is no longer the case. The .otf filename extension simply means that it is an OpenType font file that can be used on either a Mac or a PC. While TrueType fonts normally carry the filename extension .ttf, a TrueType font file that will work on either platform gets the .otf extension.

Fortunately, it no longer matters whether a given OpenType font contains TrueType or PostScript font data. Both work equally well on all computers and output devices, and all incompatibilities between the two ways of programming fonts have been ironed out. Nevertheless, if you want to, you can see what kind of font data a particular OpenType font contains. On the Mac, choose Show Font Info for a selected font from Font Book's Preview menu. On a Windows PC, this information is generally shown as part of the font file names in the Windows/Fonts folder. If this display is ambiguous, select the font file name and choose Properties from the File menu.

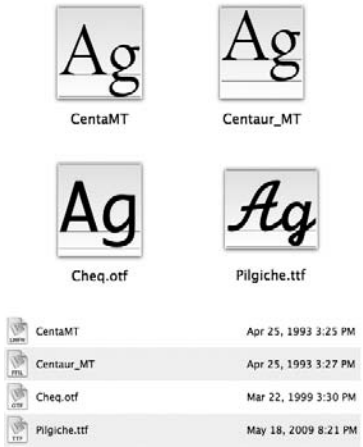

FIGURE 4.10 Beginning with OS X 10.5, Mac font icons display a sample of the typeface the font represents, as shown above. If you choose, OS X will also display filename extensions, although this may not work on older PostScript fonts, as shown on the top here. Of these two Centaur font icons, the one on the left is the outline (printer) font, and the one of the right, its bitmapped (screen) complement. For more information about the format of a font file, select the file and choose Get Info from the Finder's File menu. In the Finder's large-icon list view of the same font files below, barely visible on the icons are the legends LWFN, FFIL, OTF, and TTF.

Identifying Macintosh Fonts

The Mac icon for an OpenType font is shown in Figure 4.10, both as it appeared before OS X 10.5 and after. Starting with OS X 10.5, all font icons show a small preview of the typeface itself.

PostScript Type 1 fonts typically appear in Mac Finder windows without any filename extensions. That's because most of them predate OS X, which introduced the need for filename extensions on the Mac as a means of identifying file types. Using Get Info will reveal whether a font is a PostScript font. In the PostScript regime, each member of a font family is a separate file, so their names can become long enough that they have to be abbreviated into forms—such as OfficSerBooIta (Officina Serif Book Italic)—that may make them nearly unrecognizable. The weirdness of the names is often a giveaway.

In early versions of OS X, icons for PostScript fonts bore the label LWFN, short for LaserWriter Font (in homage to Apple's first laser printer). The icons of the companion collections of screen fonts were labeled FFIL (Font File).

These can still be seen in the Finder's large-icon list view, as seen in Figure 4.10. The FFIL label may also be applied to TrueType fonts that include embedded bitmaps for screen display at particular sizes. In certain Finder views, a TrueType font may be referred to as a *font suitcase,* a term usually reserved for a collection of bitmapped screen fonts associated with a PostScript Type 1 font.

TrueType fonts are most commonly displayed in the Finder with one of two possible filename extensions. One is .ttf (TrueType font) and the other .ttc (TrueType Collection). TrueType Collections are single fonts that contain character outline data for several typefaces. The font AmericanTypewriter.ttc, for example, can generate type in six typefaces: Light, Regular, Bold, Light Condensed, Condensed, and Bold Condensed. The icon for such a font is a small preview of the regular roman member of the family.

If you use the Finder's List view instead of the Icon view, you will have to rely on the file names alone, as the tiny icons that precede individual file names are too small to decipher. If you opt for the Column view, you have the option of displaying a preview column that for any selected font shows a typeface sample in addition to information on the font's format.

Identifying the Formats of Windows Fonts

In versions of Windows prior to Windows 7, if you look at fonts in their folders, you'll see them all identified with unique icons that distinguish PostScript from TrueType from OpenType fonts (see Figure 4.11).

In versions of Windows through XP, the filename extensions of font files are displayed in the Fonts folder. In later versions, only checking Properties in the File menu will reveal the filename extension. TrueType font names have the extension .ttf (TrueType font) or .ttc (TrueType Collection; a single font representing multiple typefaces), although these extensions can also be used for "TrueType flavored" OpenType fonts. As far as Windows is concerned, those formats are virtually identical, varying only by their character sets. Not all OpenType fonts, then, will have an .otf filename extension. Those that do will also work on a Macintosh.

If you're using Windows 7 or later, you should choose the Details view for the Font menu. Once you've turned on the option to display Font Type (by right-clicking in the column-titles bar) this view will show you each font's format plus other useful information. For most fonts you install, you can also right-click on their file names and select Properties from the pop-up menu that appears. This panel displays a range of information about each font. The Properties option is not available, however, for many of the fonts included with Windows 7.

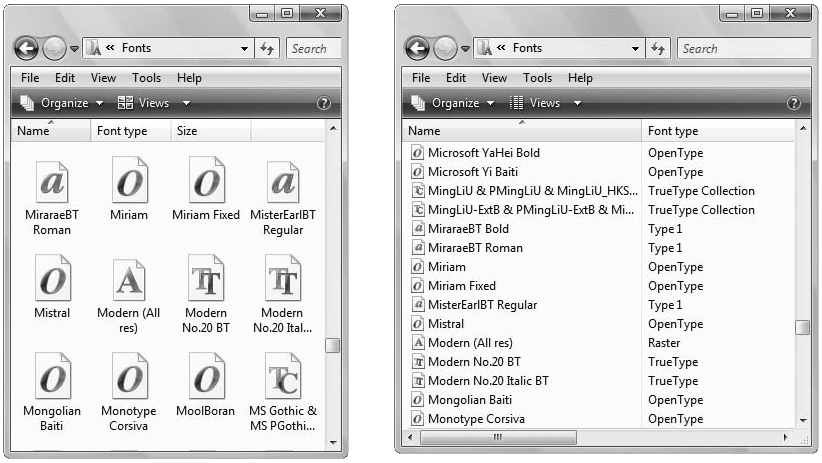

FIGURE 4.11 Versions of Windows before Windows 7 clearly identify the formats of all fonts. In the Fonts window at left, the large-icon view clearly marks the file icons with the *TT* that stands for TrueType, an *O* for OpenType, or a lowercase italic *a* for PostScript fonts. The icon with a capital *A* indicates a bitmapped font. On the right, the view has been switched to Details, which explicitly lists each font's formats under the heading *Font type*.

PostScript Type 1 fonts have the filename extension .pfb (for the font files containing the character outline data; the *b* stands for *binary*) and .pfm (for the corresponding file containing the bitmapped screen fonts and *metrics*—that is, character-width—data). Because at the time of these fonts' manufacture most versions of Windows were based on DOS (disk operating system), the length of older font file names were limited to eight characters plus a filename extension (after a punctuating dot) of three more characters. This makes the names of most PostScript fonts completely unintelligible. It's not apparent, for example, that VARG_____.pfb is actually Viva Regular. Fortunately, when they're placed in the Windows/Fonts folder (where installed Windows fonts are normally stored), Windows reads the true name of the typeface from within the font and displays it in readable form. In addition, it's common for Windows applications to indicate in their Font menu the formats of the fonts listed. Here, ideally, is where you want to know this information, and it would be preferable if all programs on all platforms performed this useful service (see Figure 4.12).

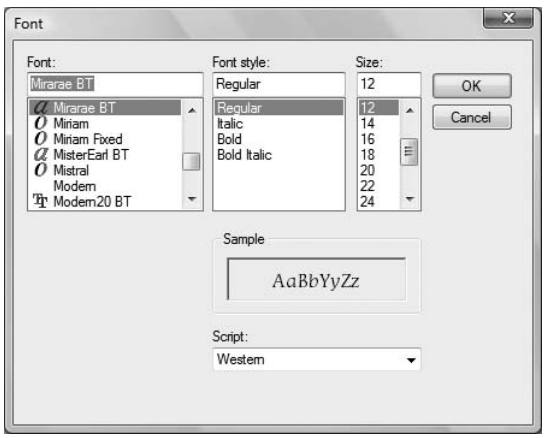

FIGURE 4.12 The Font menus of Windows programs often identify the formats of fonts with small icons. In this window, Windows Notepad distinguishes between TrueType, PostScript, and OpenType fonts in its scrolling font list. The Windows system's bitmapped Modern font appears without an icon.

The Basics of Font Management

For operating systems to be able to use them, font files need to be stored in specific places. On a Windows PC, the principal font storehouse is the Windows/Fonts folder. From here, using a command from the File menu, you can install new fonts. You can also drag font files into this folder from other sources.

On the Mac, font files are stored in Fonts folders in each of the Library folders on the computer. The fonts in the Library found in the root folder are available to all users of that Mac. Fonts stored in the Libraries of individual users (found in the Users folder) are available only to those users. The fonts in the Library inside the System folder are used by the operating system and shouldn't be touched.

Certain application programs—generally those that come with their own collection of fonts—will store fonts in their own folders, where they alone can have access to them. This is a way of assuring that other applications can't use them, as they are generally licensed to the user only for use with the host application.

Font-Management Programs

There is a breed of utility program called a *font manager,* whose job it is to help organize and manage the huge numbers of fonts that can come to populate your computer. Having too many fonts installed on your computer at the same time creates two main problems: First, it slows down your computer, which has to constantly keep track of all of those fonts. (Too many installed fonts may in fact cause your computer to freeze.) Second, it creates a Font menu that's too long to manage, requiring endless scrolling to find the font you're after. The main thing a font manager does is enable you to keep the number of fonts in your system at a minimum with very little effort.

The principal way it does this is by allowing you to install or remove fonts from active service individually or en masse at any time. You can build *font sets* to make this easier. You can have a certain set of fonts for a specific job, or a certain set of fonts that are associated with a particular program. Fonts can belong to two or more of these sets. Some font managers can automatically install whatever fonts are needed by a document that you open.

In addition, font management programs can help you organize your fonts in logical ways. Whereas an operating system would throw them in a single heap (or, worse, several hard-to-locate heaps), a font-management program can organize them according to any criteria you like: font format; historical style; text, display, or decorative use; or whatever else.

Mac OS X includes its own font-management program: Font Book, located in the Applications folder. Windows has no such utility, although there are several available from independent software developers.

Font-Editing Programs

Font-editing programs are the tools that type designers use to create fonts from scratch. They include tools to draw character outlines as well as to edit those outlines later. Font editors can be useful to the nondesigner as well, as they can add characters to a font (a digitized corporate logo, for example). They are also sometimes used by demanding typographers to improve the quality of the kerning information within a font.

Creating customized fonts can have its advantages, but its disadvantages are very serious. Edited fonts become unique fonts, and unless they have been given unique names, they can become confused with existing retail fonts. Fonts that have had their kerning information altered, for example, will cause text to compose in a unique way, and these differences are enshrined in the font, not in the document. If that document travels to a place where the custom font that created it is unavailable, it will not compose correctly, line endings will change, and whole layouts can become disrupted.

Because font files are mobile, a customized font that escapes its handlers can cause havoc if it becomes confused (or used) with its unedited forebear. Edited fonts, then, have to be employed with great care and control.

Index

Symbols

Accents

Currency symbols

Footnote reference symbols

Mathematical symbols

Punctuation

Other symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

S

T

U

V

W

X

Z