TrueType, PostScript Type 1, & OpenType: What's the Difference?

by Thomas W. Phinney

Version 2.36, December 26, 2004

Commonalities

TrueType (TT), PostScript® Type 1 (Type 1) and OpenType® are all multi-platform outline font standards for which the technical specifications are openly available. "Multi-platform" means that both font types are usable on multiple sorts of computer systems. "Outline font" means that they describe letter shapes ("glyphs") by means of points, which in turn define lines and curves. This representation is resolution independent, meaning that outlines, by their very nature, can be scaled to pretty much any arbitrary size. Depending on the particular program being used and the operating system it's run under, there may be upper and lower limits to the size the font can be scaled to, but few users will ever encounter these limits.

An outline font must be represented by the dots of the output device, whether it's screen pixels or the dots of a laser, ink-jet or wire-pin printer. The process of converting the outline to a pattern of dots on the grid of the device is called "rasterization."

When there aren't enough dots making up the glyph (such as at small sizes or low resolutions), there can be inconsistencies in the representation of certain letter features, at a single size, due to different rounding based on how the outline happens to sit on the grid. A common form of this is that the widths of the letter stems can vary when they shouldn't. Worse, key features of the glyphs can disappear at small sizes.

However, Type 1, TrueType and OpenType fonts all have a means of dealing with these inconsistencies, called "hinting." This consists of additional information encoded in the font to help prevent these problems.

Brief History

PostScript and the Type 1 font format predate TrueType by about six years (with OpenType being a much later amalgamation of the two formats). First, we had many different formats for digital fonts, none of which were standardized. Then Apple adopted the Adobe® PostScript page description language (PDL) for its Apple LaserWriter printer in 1985. This, combined with the introduction of PageMaker®, the first desktop publishing software, sparked a revolution in page layout technology.

Soon the PostScript language was adopted for use in higher-end imagesetting devices, and became the native operating mode and language of many graphics programs as well. The command structure of the PostScript language was publicly available, so it was possible for someone to build a PostScript interpreter to compete with Adobe's rasterizing software. But it wouldn't be able to interpret the hints in Type 1 fonts. This was because the PostScript font specification for Type 1 fonts, which included hinting, was not publicly available. Adobe had only released the specifications for Type 3 fonts. Type 3 fonts were a

more general format, but Type 1 was smaller, faster, and had a native hinting structure (of which see more below).

It rapidly became obvious to the major system software creators (Apple, Microsoft, and later IBM) that it was important to have scaleable font technology supported at the level of the operating system itself. This would allow much better screen display, compared to pre-made bitmaps which would only look good at a few sizes, and would be jagged at all others. So in the late 1980s, Apple developed its own scaleable font technology, first code-named Royal, and later introduced as TrueType.

Apple traded the technology to Microsoft in exchange for the latter's PostScript clone technology "TrueImage" (which was buggy at the time, and never used by Apple, although it has surfaced in various later incarnations). The TrueType specifications were made public, and TrueType was built into the next versions of the Mac and Windows® operating systems, released in 1991-92.

Adobe's preemptive response started with the release of Adobe Type Manager® (ATM®) software in December 1989. ATM scales PostScript Type 1 fonts for screen display, and for imaging on non-PostScript printers. This was followed by publication of the long-protected specifications for the PostScript Type 1 font format in March 1990.

In early 1991, TrueType for the Mac became available, followed by the Windows 3.1 implementation (the Windows scaler was and remains slightly more accurate/efficient than the Mac version, though it's nothing a normal user is likely to notice). Now, with either TrueType or ATM, Mac users (and later Windows and OS/2 users) could actually see on-screen at any size what the font output would look like.

So now there were two widely used outline font specifications, one (TrueType) built into the operating systems used by most desktop computers worldwide, and the other (PostScript Type 1) the de facto standard for publishing and the graphic arts.

But as time goes on, the practical differences begin to blur. The new OpenType format (discussed later), supports both TrueType and PostScript outlines. Support for TrueType (via Apple's TrueType rasterizer) is built in to virtually all implementations of PostScript Level 2, and is standard in PostScript 3. Similarly, Type 1 rasterizing technology is incorporated into Windows 2000, Windows XP, and Mac OS X, side-by-side with TrueType and both flavors of OpenType. Indeed, Apple's new Japanese system fonts provided with the OS are in OpenType form, albeit with some Mac-specific additions.

Technical Differences

The first difference between TrueType and PostScript fonts is their use of different sorts of mathematics to describe their curves. OpenType fonts can have either kind of outlines, with their respective advantages and disadvantages.

Some articles have said that TrueType fonts require more points than PostScript, or that they take longer to rasterize because the math is more complicated. In fact, the math is simpler (quadratics are simpler than cubics). Although a few shapes take fewer points in TrueType than in PostScript (a perfect circle takes twelve points in PostScript vs. eight in TrueType), in practice the shapes in real-world fonts all tend to take more points in TrueType. So it's true that most fonts will end up using more points in TrueType, even if the kind of mathematics used to describe the curves is simpler.

The primary advantage of TrueType over Type 1 fonts is the fact that TrueType has the potential for better hinting. Mind you, PostScript Type 1 hints handle a lot: vertical and horizontal features, overshoots, stem snaps, equal counters, and shallow curves ("flex"). Several of these can have a threshold pixel size at which they activate.

However, TrueType hints can do all that Type 1 can, and almost anything else, as defined by the very flexible instructions of the TrueType language. This includes controlling diagonals, and moving specified points on the glyph outlines at specific arbitrary sizes to improve legibility. This ability to move points at a specific point size allows font production staff to hand-tune the bitmap pattern produced by the outline at any specified size.

Or at least it used to; more recent divergences in TrueType rasterizing between different players (including Apple and Microsoft) make this a little more uncertain. Worse, some of the most advanced

sub-pixel rendering (Microsoft's ClearType, Adobe's CoolType and Apple's OS X font rendering) ignores a lot of the high-end TrueType hinting instructions.

This difference in hinting philosophy is really symptomatic of a larger philosophical difference. PostScript uses "dumber" fonts and a "smarter" interpreter, while TrueType uses relatively smarter fonts and a dumber interpreter. This means that PostScript hints tell the rasterizer what features ought to be controlled, and the rasterizer interprets these using its own "intelligence" to decide how to do it. Therefore, when someone upgrades their PostScript interpreter, the rasterization can be improved.

Contrariwise, TrueType puts all the hinting information into the font to control exactly how it will appear when rasterized. Some TT aficionados prefer to call TrueType hints "instructions," partly in reference to the full-featured nature of the TrueType programming language, but also to clarify the role of this information. As Jelle Bosma of Agfa Monotype says, "I don't *hint* at what I want to happen—I *tell* the font what to do."

Thus the TrueType font producer has the potential for very fine control over what happens when the font is rasterized under different conditions. However, it requires serious effort, expertise, and high-end tools for a font developer to actually take advantage of this greater hinting potential. Also, making major changes to the TrueType rasterizer while displaying existing fonts at their best has proved difficult to manage so far.

Until recently, the other advantage of TrueType was that it was the font format supported directly by the Mac and Windows operating systems, while Type 1 required an add-on. These operating systems will rasterize TrueType fonts for the screen, and send them to printers, whether as bitmaps or in some font format the printer understands.

Scaling either PostScript fonts, or OpenType fonts with PostScript outlines, on Mac OS 8/9 and Windows 95/98/ME, requires the Adobe Type Manager (ATM) software, which handles the rasterizing to the screen, and rasterizes or converts the fonts for non-PostScript printers. (Technically, Mac users don't require ATM to use PostScript fonts on PostScript printers, but ATM is required to display the font accurately on screen at arbitrary sizes.) ATM is freely available: the "Light" version is a free download from Adobe's Web site, and also comes with many Adobe applications.

However, in Windows 2000 and XP, and Mac OS X, the PostScript Type 1 and OpenType CFF support is built in, just like the TrueType support has long been. So this former advantage is rapidly vanishing.

A smaller, but consistent, advantage of OpenType and TrueType has to do with the physical storage of the fonts. OpenType and TrueType fonts have all the data in a single file. PostScript Type 1 fonts require two separate files: one contains the character outlines, and the other contains metrics data (character widths and kern pairs). On the Macintosh, Mac OS 8.1 and earlier requires Type 1 fonts to have not only the outline font, but also a bit-mapped screen font in at least one size, which contains the metrics data. For Windows systems using PostScript, a "PFB" file contains the outlines, while a "PFM" file carries the metrics.

The system-independent "AFM" metrics file can be converted to a Windows PFM file upon installation by ATM (if accompanied by an INF file), or can be used by a font editing program along with the outline to create a screen font for the Mac that includes any kerning pairs in the original.

On the other hand, PostScript's pair of files are often smaller than TrueType's single file. The size difference ranges from only a 5% savings for an average font, to as much as a doubling of size for TrueType fonts that actually have extensive "hinting" instructions.

Also, most high-end output devices use PostScript as their internal page description language. PostScript fonts can be sent directly to these devices. It used to be the case that TrueType fonts were either downloaded as bitmaps or required that the TrueType rasterizer be downloaded as a PostScript program, which slowed printing a bit.

More recently, many PostScript Level 2 printers, (and all PostScript 3 printers) have the TrueType rasterizer in ROM, built in. However, with some Windows printer drivers the user must change the

printer driver settings in software to take advantage of this feature (downloading TrueType as "Type 42," which is basically a PostScript wrapper around the TrueType data).

Further Practical Differences

Many of the theoretical advantages of TrueType are not actually realized in most commercially available TrueType fonts. PostScript backers point to a number of problems that still make PostScript fonts a better solution for many users. Besides the above-mentioned issue of the language of the output device, there are four other practical issues that even the score for PostScript:

First, at present many of the commercially available TrueType fonts one sees at the software megamart are of poor quality, coming in "zillion-fonts-for-a-buck" collections. Many of these fonts were originally shareware or public domain PostScript fonts, and were converted to TrueType using some basic automatic utility. The outlines and hinting are no better than they were in the PostScript versions, and will suffer slightly in almost any automatic conversion. Usually in the case of extremely cheap collections, they weren't the best quality* PostScript fonts even before conversion to TrueType.

Of course, TrueType backers point out that often these fonts were available before; it's simply the availability of a universal font scaling technology that makes discount fonts for the masses practical, and of course they are more likely to be released in the most widely available format.

Second is the issue of easy-to-use tools. On the plus side, there is a retail font editor with native TrueType support (FontLab), as well as Microsoft's Visual TrueType (VTT) hinting tool. However, regardless of the specific tools used, achieving first-class hinting in TrueType currently requires intensive manual coding on a glyph-by-glyph basis. This requires substantial time and expertise on the part of the person doing the hinting.

As a result, high-quality TrueType fonts are currently only available from a handful of vendors, and only a minority of even those fonts really exploit the potential of TrueType hinting.

Third, TrueType's hinting advantage only matters when hinting matters: when outputting to low-resolution devices, or for screen display. The increasing, widespread use of 600 dpi and better laser printers makes this less critical for print work. Although one might think that the increasing importance of screen displays for so many purposes—including multimedia, the Internet, and electronic books—makes hinting more important, it turns out that on desktop and laptop computers, the opposite is true. The new screen display techniques mentioned earlier make hinting much less important. Indeed, superhinted TrueType fonts tend can look worse than their more moderately hinted brethren, when such techniques are in use.

Fourth, PostScript has some advantages simply from being the longer-established standard, especially for serious graphic arts work. Service bureaus are standardized on, and have large investments in, PostScript fonts. Most of the fonts which have "expert sets" of old style figures, extra ligatures, true small capitals and the like are in PostScript Type 1 format.

Although most major vendors have TrueType fonts, not all offer their entire libraries in both formats. Agfa Monotype, Linotype and Bitstream have their entire libraries in both formats, while Adobe has but a handful of TrueType fonts. Given the current state of the tools, although a simple conversion would be easy, it would take a concerted effort of many years to convert all the major vendors' font libraries to TrueType if they also wished to enhance the quality.

Interoperability

Another often-raised issue is the story that some PostScript devices, particularly imagesetters, have problems either with TrueType fonts in general, or especially with mixing TrueType and PostScript on the same page or the same line. This is mostly an historical issue. Recent implementations of TrueType in operating systems, and newer Adobe PostScript

* What do I mean by poor quality? Incomplete character sets, inconsistent stem weights, improper outline construction, excess points, inadequate or improper hinting, inconsistent spacing, poor or nonexistent kerning, and many other factors.

interpreters, have resolved what few problems there were in the early 90s.

According to Dov Isaacs, Adobe's Manager of Quality Assurance, Printing & Systems Division in the early to mid-90s, "regardless of whether you are on a Mac or a PC running Windows 3.1 or above, you can mix TrueType and Type 1 with the caveat that you should never have both TrueType and Type 1 fonts with the same exact names on the same system." Indeed, having any two fonts with identical menu names or PostScript font names can confuse the operating system or your applications, with unpredictable results.

Also, if using Windows, one may find that metrically-similar PostScript fonts get substituted for the Windows TrueType system fonts at output time: Times New Roman® becomes Times® Roman, and Arial® becomes Helvetica®. Getting the same font on the actual output can be guaranteed by changing printer settings in the printer control panel, to ensure the TrueType system fonts get used. Hackers can also try editing the WIN.INI file on the computer that is doing the printing (whether to device or file). Delete the relevant lines in the font substitution section, so that the TrueType font used on-screen is also sent to the output device, rather than a printer font being substituted. On Windows NT/2000/XP, Registry settings control the same behavior. Alternatively, get a scalable version of the font used in the printer, and use it instead of the system fonts.

When dealing with fonts on the computer's side, one needs to be careful about deliberately substituting Arial for Helvetica and Times New Roman for Times, or vice versa. Although the basic spacing of the substituted fonts is identical, their kerning pairs are not. This can cause text to reflow if one switches between two different-but-almost-the-same fonts on the computer doing the typesetting, if the program supports kerning pairs (graphics and DTP programs, and some better word processors). In situations in which exact line breaks are not critical, or applications in question do not use kerning, problems are unlikely.

One actual, but rare, source of problems is not inherent in TrueType, but a result of the fact that rasterizing TrueType can require a bit more RAM in the raster image processor (RIP) than rasterizing PostScript—primarily in much older PostScript Level 1 rasterizers when the TrueType rasterizing program must be downloaded. If the RIP has barely enough RAM, it's possible that this could push it over the edge.

A more common source of problems is that some non-Adobe "PostScript-compatible" imagesetters do not support TrueType properly.

Service bureaus and printers are notoriously conservative about these sorts of thing (understandably, since any delays or problems can cost them and their clients money); your best bet is to consult with them, and if they warn of potential problems, test something complex with a mix of font formats for future reference.

Converting TrueType & PostScript

Mathematically speaking the quadratic B-splines of TrueType are a subset of the cubic Bézier curves of PostScript, so it's possible to convert TrueType to Type 1 without loss of accuracy. And if enough points are used, one can convert in the other direction with minimal loss.

But this is only true if the same design space is used. Most TrueType fonts are designed on a 2048-unit grid, while PostScript Type 1 fonts typically use a 1000-unit grid. Although neither of these measurements is required, if the conversion does choose to change the measurement basis (or "em-square" in fontspeak), there will likely be changes in the outlines due to rounding.

More importantly, hinting information does not directly translate in either direction between the two formats.

Within these limitations, a variety of retail tools (both font editing programs and dedicated conversion utilities) can convert between PostScript Type 1 and TrueType. For a casual user, the results are likely to be acceptable. As of this writing, there are no shareware or freeware utilities that perform such conversions.

Multiple Masters

The PostScript Type 1 multiple master (MM) format is an extension of the Adobe Type 1 PostScript font

format. Essentially, it allows two design variations to be encoded as opposing ends of a single design axis. Afterwards, any in-between state (an "instance" in MM-speak) may be generated by the user on need. Thus, an MM font could have a "weight" axis which has an ultra-light master and an extra-black master, allowing any conceivable variation in between. And this is only one possibility; almost any two design extremes could theoretically be put on a multiple master, as long as their Bézier control points can be matched up to allow interpolation.

Multiple axes are also possible, but (in all implementations, though not technically required by the format) each additional axis doubles the number of master fonts that must be created, because each possible extreme must be designed separately. Imagine a dimensional space, with each corner requiring a master. Thus a three-axis MM (a cube) must have eight master fonts; a four-axis font (the practical maximum) would need sixteen master fonts, which is one reason nobody has released one yet.

The primary uses to which MM technology has been put are: weight (light to bold); width (condensed to extended); and optical size (text to display). A few MM fonts experiment with other forms, such as the existence or type of serifs. All of these adjustments can be done by cruder means, by creating separate fonts, or even just ignored; but multiple master fonts allow typographically aware users to create the precise, desired typeface in a more refined fashion.

Multiple master fonts come with a variety of predefined font instances, which meet many users' needs, and make it unnecessary for some users to create further instances.

Unfortunately, it can be inconvenient to get to MM instances other than the predefined ones. Most of the time, the user must use ATM to instantiate each additional font variant in order to make it available to the system. There are a few exceptions: Microsoft Word 6 and higher, and QuarkXPress 3.3x and up, support direct creation of MM instances on the fly by typing the exact name of the instance (easy, but hardly obvious). PageMaker 6 and better also has integrated support for creating and using MM instances, as does QuarkXPress 3.3x, via an included extension. Only Adobe Illustrator® 7 through 10 went so far as to allow direct manipulation of MM axis sliders "live" on text. Adobe InDesign® doesn't have this, but does automatically use the correct optical size instance.

There are a few older devices with implementations of PostScript level 1 that can't handle MM fonts, notably Apple's Personal LaserWriter NT, the HP LaserJet IIID, the PostScript cartridge for the HP LaserJet IIP, and the TI microLaser PostScript series. Additionally, some older PostScript clones may have problems with multiple master fonts.

Because with most applications it is inconvenient, and because many users are unfamiliar with MM technology, it often makes more economic and marketing sense to release a font set as multiple separate fonts, even if it was designed using multiple master-style interpolation. Examples of this trend include Jonathan Hoefler's reworking of Didot, and most of the fonts released by Adobe in the last five years.

Fewer than 50 MM fonts have been released by major font vendors—and more than half by Adobe. Using multiple masters also requires that the user have Adobe Type Manager (even in Windows 2000 and XP), but this is a near-necessity for PostScript fonts in many environments, anyway.

In October 1999, Adobe announced that it was ceasing development of new multiple master fonts, citing the lack of application support, engineering costs to support MM technology, and Adobe's desire to concentrate its resources on OpenType. In 2002-03, Adobe phased out sale of multiple master fonts as equivalent OpenType versions became available. Illustrator CS, released in late 2003, removed the MM creation slider control. In September 2004, Adobe announced that support for MM fonts would end at the end of the calendar year, though this would be coupled with a special discount for registered owners of Adobe MM fonts on the closest equivalent OpenType fonts.

Unicode

In discussing other extensions to TrueType and PostScript, it is helpful to first discuss Unicode, since several of them support or are based on this technology. Unicode is an international standard for rep-

resenting a broader character set using multi-byte encoding for each letter. This allows the encoding of thousands of characters instead of 256: essentially all the characters for every language in the world, each with a unique ID.

However, the Unicode specification only covers differences that have a linguistic impact, such as accented characters. It does not deal with typographic niceties such as unusual ligatures, old style numbers, or small caps. To paraphrase Chuck Bigelow, it may seem like a metaphysical distinction, but Unicode is a character encoding, rather than a glyph encoding. The result is that simply adding Unicode capability is very useful for non-English or multi-lingual typography. However, it does not, in and of itself, aid in dealing with the typographic issues addressed by, say, GX/AAT or OpenType (discussed below).

There are alternatives to Unicode, such as Apple's initial GX solution of multiple single-byte encodings per font, and Adobe's CID technology. However, most such alternatives are stopgaps; both Apple and Adobe have added Unicode support to their technologies (Apple Advanced Typography replacing GX, and OpenType with CID replacing Type 1 with CID).

Unicode character encoding is directly supported by Windows NT, 2000 and XP. The Mac OS had the beginnings of Unicode support as far back as OS 8.5, and significant support in Mac OS X, though many major Mac applications still do not support Unicode. Microsoft's Office 2004 for Mac finally has Unicode support, and Adobe's Creative Suite applications (InDesign, Illustrator and Photoshop) do as well—though independently of the OS. Additionally, OpenType (see below) is directly based on Unicode, and thus operating systems and applications that fully support OpenType will support Unicode in the process.

National Language Support & WGL4

Windows 95/98 and ME did not fully support Unicode, but have a less universal approach called National Language Support. NLS is accessible in foreign-language versions of Windows 9x, or if a user installs Multi-Language Support. One can then make use of TrueType fonts with more than the usual 256 glyphs of WinAnsi. For convenience, and to help preserve compatibility with older programs, the user's selected language setting determines which two-hundred-odd glyphs are accessible from the keyboard (the correct ones for the chosen language, assuming they're in the font).

The Windows Glyph List 4 (WGL4) character set is a specific NLS set of some 652 characters, which include all the characters for every European language. This means all the usual regular and accented Latin characters, more accented Latin characters for central Europe and the Baltic countries, plus Greek, Cyrillic, Turkish, a host of accented characters, and IBM Linedraw thrown in for good measure. The basic Windows system fonts (Arial, Courier, Times New Roman) have all been upgraded to WGL-4 (or more), as have a number of other TrueType fonts distributed by Microsoft, including Verdana, Tahoma, Andale Mono, Impact, Palatino Linotype, and the Microsoft version of Franklin Gothic.

OpenType

This 1996 Adobe/Microsoft initiative surprised industry analysts. OpenType puts either PostScript or TrueType outlines in a font, with tables including the current TrueType tables and additional tables for advanced typographic features. Non-technical people might think of it as a common "wrapper" based on the existing TrueType structure. Applications—and most operating system functions outside of font rasterizing—will no longer care which type of font is in this "wrapper." In some senses, the OpenType approach to putting TrueType and PostScript in a common wrapper is very much like how PostScript Type 1 is supported in a GX/AAT environment.

As part of the deal, Microsoft and Adobe licensed the TrueType and PostScript font technologies to each other, and pledged an end to the "font wars"—the longstanding debate over which format was better.

The representation of Type 1 font software in an OpenType font uses Adobe's Compact Font Format (CFF) with Type 2 charstrings, which may also use subroutines for added compression. This is a dramatically more compact representation of the same information as Type 1. Indeed, Adobe says a Type 1 font

converted directly to subroutinized OpenType CFF, without added glyphs and features, is 45% smaller on average. (Adobe had started work on CFF in late 1995, initially for use in PostScript 3 printer ROMs, but it has found much wider use in Adobe Acrobat and in OpenType fonts.)

The OpenType format supports features equivalent to most of the advanced features of existing TrueType and PostScript formats, such as Adobe's CID technology for Asian fonts, and extended multilingual character sets. However, multiple master fonts are not part of the OpenType specification.

OpenType fonts allow extended character sets beyond the usual 256 which can be encoded in standard PostScript Type 1 fonts. These can be alternate letterforms, or those characters formerly included in "expert sets," additional languages, or whatever the designer desires.

The key additional typographic layout features in OpenType are supported by means of additional "tables" of information in the fonts, which specify how the glyphs are modified by application of features. For example, real (specifically designed instead of simply scaled) small caps can be built into the font, and feature tables could define the relationship of these small caps to both regular caps and lowercase letters. Similarly, feature tables can define such things as ligatures, swash characters, alternates, etc.

These tables are the basis of automatic glyph substitution. Substitution need not be one for one; one glyph can be substituted for several (such as the f-f-i ligature, or many Arabic characters), or multiple glyphs can be substituted for a single one. Glyph substitution can be context sensitive, and/or activated by explicit user activity.

There are several advantages of this over the currently available "expert sets" and "alternates." First, the user's font menu isn't cluttered with supplemental fonts. Second, there can be kerning between glyphs that might otherwise have been in separate fonts. Finally, with an appropriately savvy application, the user can turn on ligatures, small caps, or oldstyle figures, much like bold or italic styling, without switching fonts.

Although Seybold analysts initially reported on OpenType as a victory for Microsoft and TrueType (by them getting legitimacy in publishing), that's a pretty narrow view. In the broad view, everybody wins. Microsoft may indeed finally be getting greater TrueType acceptance in the high-end publishing market. Adobe got PostScript font outline support at the system level in Windows, potentially making the Adobe type library more accessible to a broader range of potential buyers. But best of all, end users win by getting a single standard for advanced features and cross-platform fonts, and eliminating one of the largest remaining hassles for document transfer between Macintosh and Windows computers.

Although Apple ships Japanese system fonts for Mac OS X in OpenType format (with PostScript outlines, and some added AAT tables), OS X does not yet have native support for any OpenType features beyond imaging the fonts, and in 10.2 and later kerning for basic western characters in carbon—but not cocoa—applications.

Meanwhile, Adobe released its last new Type 1 font in 1999, and has converted the entire Adobe Type Library to OpenType (over 2000 OpenType fonts). Other foundries have been slower to move to OpenType, but movement accelerated in 2003 and early 2004. URW++ and Elsner + FLake have converted their entire libraries, over 2000 fonts, and Linotype has shipped over 200 of its own, aas well as reselling Adobe's. Many others are shipping some OpenType fonts, including Agfa Monotype, Bitstream, House Industries and Émigré. Freeware and shareware font makers are even offering a couple of dozen OpenType fonts.

Among publishing applications, so far Adobe InDesign®, Photoshop® 6 and higher, and Illustrator CS and later support OpenType layout features. The level of support varies between the applications somewhat, and tends to be greater in newer versions.

Initial Microsoft feature support across the Microsoft Office applications has been on Windows only, and solely for those features which are necessary for language support, such as contextual substitutions for Arabic—and only in the languages which require them (although Word 2000 and later will do contextual substitutions for Arabic, as of this writing it won't do them for English).

GX & AAT Fonts

Another attempt to enhance these typographic niceties has been Apple's GX/AAT fonts. This font technology, born in 1991, was first part of the QuickDraw GX printing/graphics technology, which was later abandoned by Apple. However, the font part of GX has renewed life as "Apple Advanced Typography" or AAT in 1998. AAT is in turn an element of "Apple Type Services for Unicode Imaging" or ATSUI. Both AAT and ATSUI, at least in basic form, are part of Mac OS 8.5 and higher, including Mac OS X. In Mac OS 9 and beyond, all the system fonts are AAT fonts.

How do AAT fonts work? AAT supports TrueType fonts, and other outline formats that use the TrueType table structure. Like OpenType, AAT fonts also allow extended character sets beyond the usual 256 allowed by standard PostScript Type 1 fonts. They are referenced by tables, like OpenType approach, although the AAT tables function a little differently, being "state tables" rather than simple lookups.

The GX/AAT Line Layout Manager is a bit of system software that interprets and manages all this additional information encoded in the font's tables to do useful things, such as accessing the small caps mentioned above, automatic intelligent ligature substitution, or optically aligning the edges of text based on the actual shapes of the letterforms rather than the outside of the character bounding box.

TrueType GX/AAT fonts can also be designed as "variation fonts," similar to multiple master fonts with design axes. However, TrueType AAT also has greater flexibility in the use of these axes.

Developers were initially very reluctant to make GX/AAT applications. One reason is that it is only available for the Macintosh, and most major layout software is actively seeking cross-platform compatibility; therefore the vendors are loathe to adopt a "standard" that doesn't have a counterpart for Windows (or Linux, or any other systems they may support).

A second barrier has been that AAT as a layout model has historically tried to take over line layout, an area in which high-end layout applications have put considerable effort into adding features and value for the end user. The makers of such applications would be understandably reluctant to abandon their previous hyphenation and justification capabilities (for example) in favor of AAT capabilities which are delivered "free" to the lowliest word processor which chooses to support AAT.

This barrier may be going away, however. Apple says it is moving towards making AAT functions accessible to applications without requiring them to give up all line layout control.

Another barrier was removed by Apple back in 1988, in separating out the GX imaging/graphics model. Users can now use AAT-savvy applications without installing system software which is incompatible with other major graphics applications.

However, none of the biggest software vendors have released any applications which are AAT-savvy. There were about a dozen programs that offered some degree of support for AAT in its former GX guise, including two page layout programs, Uniqorn and Ready-Set-Go 7 GX, and LightningDraw, a drawing package. Such applications need rewriting to work in current Mac OS versions (with AAT but without GX), but so far only Ready-Set-Go has received such treatment. The most prominent GX application was Multi-Ad Creator 2, but the most recent versions are no longer based on GX.

On the plus side, it appears that most or all Mac OS X applications written as "cocoa" applications (see Apple's site for details) use the NSText library for managing text, giving them automatic "free" access to AAT functionality. Most notably, Apple's Keynote presentation program and the Stone Create graphics application have full AAT support. So far, the only applications that are written for "cocoa" are Mac-only applications written from scratch for OS X. Still, this is likely to yield even more application support for AAT. Additionally, it may be theoretically possible for "carbon" applications to also support AAT, though no major applications seem to fall into this category.

Font foundry support for AAT has been very irregular. Some type foundries that originally released or planned to release GX fonts either withdrew them from circulation, or failed to release the announced fonts. However, sztarting in late 2003 there has been some interest in AAT from type foundries. Some

have investigated making "hybrid" fonts with both AAT and OpenType layout tables.

Thus, in Apple's current operating system strategy, GX proper is dead, but GX typography as AAT is still being pushed. Apple's support of Unicode in AAT and integration of AAT into the Mac OS is increasing support for AAT, but its long-term prospects are still unclear.

What about "dfonts"?

A source of additional confusion for Mac OS X users is Apple's introduction of the "dfont" format. This is essentially a repackaging of a "traditional" Mac OS TrueType font, placing the resource fork information into the data fork—dfont is short for "data fork font."

Apple introduced the dfont format to meet internal OS production needs. OS X is based on Unix under the hood, and Apple requires that all the system components be normal Unix files, meaning they can't have resource forks. Apple has been very clear that dfonts are an internal, Apple-only system thing, and that they don't expect anyone else to make dfonts.

The main problem with dfonts is one that already existed for Apple's previous TrueType system fonts: many of them have names that conflict with pre-existing Type 1 fonts. More detail on how to deal with this, and other issues around fonts on Mac OS X, can be found in Apple's white paper on font issues, at *http://www.apple.com/pro/archive/creative/fonts*.

What Does the Future Hold?

One thing that drives acceptance of some Unicode-based solution, is the needs of international markets. As mentioned earlier, Unicode is a broader and more complete basis than any other for multi-lingual computing. This is important to both operating system companies such as Apple and Microsoft, and to vendors (such as Adobe) of printing systems, applications and fonts for international markets.

Windows 2000, Windows XP and Mac OS X have built-in support for all three font formats. Adobe has shipped its entire type library, over 2200 fonts, in OpenType, and three of Adobe's flagship applications, InDesign, Illustrator and Photoshop, all support some OpenType layout features and use Unicode under the hood.

OpenType may be a savior in the font wars, thanks to its combination of features, cross-platform functionality, and the companies backing it—but applications must be updated to take advantage of its more whizzy features. Although existing font libraries can easily be converted without added features, it is only by the merging of supplemental fonts and the laborious addition of new features, as Adobe has done, that the greatest value can be added to a converted library. As of this writing, at least two dozen vendors are developing OpenType fonts for western languages. Although there are many OpenType fonts now available, the huge installed base of older formats means there will still be occasions when users have to choose between PostScript and TrueType—and even within OpenType, both outline formats are available.

As we have seen, there are definitely situations in which one format or another may be desirable, such as when particular expert sets are needed (more commonly available in PostScript fonts, or integrated in OpenType), when TrueType doesn't work on a particular older imagesetter, when maximum legibility is needed for screen display (the best TrueType and TrueType-flavored OpenType fonts), when easy access to advanced typographic features is needed (from full-featured OpenType fonts), or cross-platform font files are needed (OpenType again).

Despite these distinctions, the relative advantages of each format are often exaggerated by their boosters. OpenType has new capabilities; but most of these are not yet widely supported in applications. In practice, most users can usually use any of the three formats, and mix them, without worrying a great deal about the differences—and said differences, except for enhanced OpenType features, are usually transparent to the end user.