# COMPATIBLE, TRANSPORTABLE TYPE AND PAGE DESCRIPTION LANGUAGES

René L. Delbar*

Abstract: This paper deals with the actual situation and future directions for page description languages used to drive various output devices, display methods for representing pages on workstation screens, and the associated techniques for font encoding and character rendering.

Recent developments in the areas of output device and display screen control have opened roads towards better solutions for a close match between screen and final output. At the same time, they may lead to parallel and mutual incompatible directions, weakening the chances for broad document interchangeability, and if nothing else, seriously confusing the end users.

To start, what is a "page description language" (or PDL in short)? During the design and layout phase, the application software stores the definitions for each page element, together with their placement and interaction information, in some internal database. Essentially, a page description language is a vehicle to carry the page data from the workstation down to the output device, where the page will be imaged (Figure 1). During this transfer, the description of the page is translated into a PDL command stream.

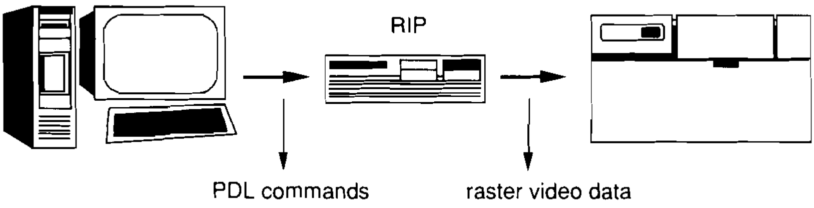

Figure 1. Typical output data stream.

The next step in the output process is handled by the "raster image processor" (or RIP). This dedicated computer system executes the individual PDL statements; it generates a bitmap of pixel map representation of the page, matching the size, resolution and other imaging requirements of the output device. Finally, the page data are transmitted to the marking engine, typically as a raster video signal synchronized with the physical scanning process. It is clear that both the application program, generating the PDL command stream, and the RIP executing these instructions, must be based on the same graphics imaging model.

* AGFA Compugraphic, Wilmington, Mass.

The model represents their common view of the world, and describes the basic concepts and functionality for constructing composite structures. Operations such as filling, clipping, and transparency have to be uniquely specified.

Much in the same way, workstation and RIP must share common methods for handling text and fonts. As a minimum, both sides have to agree on font references (typeface names, character metrics, etc.). Ideally, the application and the output device will use exactly the same graphic environment, i.e. imaging model, graphics and text command language, and font representation. This situation guarantees as close a match between display and final hardcopy as the obvious differences in resolution and color/tone scale will allow.

In practice however, most publishing programs incorporate their own imaging model, and will translate a private set of graphic primitives into the appropriate command stream for each of the supported output devices. This may result in some discrepancies between the graphics capabilities of the program and the output device. Some application features may only be available in conjunction with selected groups of output devices, and not all functionality inherently offered by a PDL or RIP may be accessible from the program. As both the workstation system environments and the output devices further converge towards a limited set of windowing systems, imaging models and PDLs, it can be expected that compatibility – or at least consistency – will improve.

Calling Adobe Systems' PostScript the dominant standard page description language within the desktop publishing and electronic prepress worlds, is almost an understatement. PostScript's success is based on its broad, general imaging model, designed to be able to describe any page that can be imagined. To reach that goal, PostScript provides seamless integration of text, structured computer graphics, and continuous tone images. Its strong typographic capabilities are supported by a large and continuously growing typeface library, with contributions from all major traditional type libraries. Finally, PostScript RIPs have been made available for a broad range of output technologies: all of them accept basically the same device independent page descriptions.

Today, the term "PostScript" covers more than just the original printer page description language. Encapsulated PostScript Format (EPSF) files are used increasingly to exchange graphic documents – ranging from line art illustrations to complete pages – between different application programs and/or different publishing workstations. For some of these computer systems, Display PostScript™ is the heart of the screen display and windowing subsystem. The Adobe Type Manager software implements the PostScript font scaling and rendering technology on yet other systems.

All of the above was not accomplished overnight, and there is still a lot of criticism with respect to PostScript. To better appreciate the why's thereof, it is important to distinguish the definition of the language itself (including the underlying imaging model) from the various product implementations.

The hardware of the Adobe RIPs was initially designed for relatively cheap plain paper printers, to be used by relatively unsophisticated application programs. As software developers and desktop publishing users started to discover the graphics potential of PostScript, and as the performance demands of output devices (be it resolution or speed) went up, PostScript RIPs seemed slower and slower. Although recent RIP designs are based on significantly more performant hardware, increases in application sophistication, user experience (and expectations!) and output device speeds will continue to consume whatever horsepower is added to the RIPs.

Early PostScript drivers did not exactly do a good job either, when converting from the application's graphics model into the PostScript world, which was another reason for performance problems. Output times for pages containing the same elements, on the same output device, but using different application programs, still vary over a wide range. As PostScript is gaining acceptance, driver implementations have improved, and application imaging models have been adapted to more closely reflect the output capabilities.

It also remains a remarkable fact that most people will associate PostScript with output from Apple Macintosh computers. In reality, virtually all Mac programs output a QuickDraw display list, that is translated into PostScript by the LaserWriter driver software...

On the quality side, the first debate came on the rendering of typefaces, or more precisely, how close the character shapes at large point sizes and/or at high resolutions match with the original design of the face. It is obvious that the early PostScript RIPs had problems rasterizing some fine and complex characters, and that these were sometimes avoided by minute changes to the character outlines. This process of "regularization" will be discussed later. As the quality of the RIP implementations improves, and the font designers develop a better understanding of their capabilities and limitations, this situation has significantly changed.

Perhaps the single biggest area of quality concern has to do with the halftone screening techniques currently used within the Adobe RIPs. What used to be more than appropriate for 300 dpi laser printers, may not match the reproduction requirements of precision, high resolution imagesetters.

The problem is most prominent when making color separations, as the standard Adobe PostScript screening techniques result in a limited choice of screen frequencies and angles. Moreover, the values are different from what the color trade is traditionally using, and that "non-conventional" character alone became the basis for ardent discussions. The whole issue is complicated by the fact that the battle is fought in the desktop arena, where there is a general lack of understanding of color issues and applications to begin with.

There should be no doubt that all of these problems will be solved in the coming years. As the size of quality demanding markets grows, and users become fully aware of the requirements for quality, performance and control, they will be more prepared to pay for an appropriate RIP implementation , and specific product offerings targeted at these markets will appear. There will be faster RIPs, using advanced hardware concepts such as RISC computers, large cache memories, and graphics co-processing. The improvements in processing power will be matched by faster internal storage and more efficient data communication channels, to make sure that the incremental power can be used to a maximum.

Large volume implementations for product categories with little variance (such as printers) will continue to be based on dedicated hardware implementations. The more optimal use of the computer resources and the lower build cost per unit will compensate for the extra engineering effort.

Implementations based on standard PC or workstation platforms will also de developed. In some cases, the goal is to further lower the cost of the RIP for environments with low output volumes, such as personal printing. For other applications, aimed at the high end or at specialty markets, lower product volumes and access to a wide price/performance range will overrule the extra hardware cost and system overhead.

There will be RIP versions that can drive multiple output devices using different imaging technologies, such as a silver based imagesetter and a plain paper proofer, or a color and a black and white device. There will be improvements to halftone screening methods, commonly involving hardware support. Data compression techniques will allow faster transmission and more efficient local storage of bulky image data. Procedures for color control and color calibration will extend the goal of device independent document exchanges into the color arena.

Full document interchangeability, after all, is the major objective pursued by the user community. This implies absolute compatibility between the sending and the receiving system for graphics as well as for text. In addition, true "what-you-see-is-what-you-get" (WYSIWYG) conformance between the workstation display and the final hardcopy is essential to support layout, composition and design decisions while working interactively on the screen.

There is an ongoing evolution in the font encoding and rendering area, bringing different and competitive solutions to various parts of the publishing system. The users however want to protect their considerable investment in fonts. They do not wish to replace their entire font library as they move on to newer output devices – potentially from a new vendor – nor do they like multiple font sets supporting displays and output devices. "One font for all" is obviously their preferred choice.

Finally, the desire to use the text character as a true design element is heard increasingly. The closed font technology of the previous system generations has limited the capabilities for the end user to modify character outlines, make partial changes to fonts, or even to create complete user defined fonts.

The font revolution started with display systems switching from using character bitmaps towards character outlines. An outline is a mathematical description of the character's shape; it fully describes the character within a device independent coordinate system. The outline may be constructed using straight line segments, arc segments, or higher order curves. Using less complex building blocks increases the difficulty to precisely match the intricate character forms. As a result, less smooth curves with visible discontinuities at the joints will become apparent at larger sizes. This can be compensated by adding more shorter curve segments to the outline, thereby increasing the description data size.

Bitmaps are small images representing characters at a specific size, orientation and resolution. The bitmap data indicates which pixels will be turned on to display the character on the screen or on the output page. Since the size of the pixels is fixed by the display technology, only a limited number of them will be available to show small characters at low resolutions. That results in very crude approximations of character shapes.

More important, character bitmaps are not scalable, i.e. a bitmap calculated for one display size cannot be used to represent the character at any other size, without serious deformation or loss of quality.

Of course, any digital reproduction system will ultimately use character bitmaps. The important difference is whether these bitmaps, for various sizes and resolutions, will be pre-calculated (and perhaps hand edited) and stored permanently within the system, or will be generated from the outline representation on-the-fly, at the proper size, angle and resolution, when needed.

It was mentioned earlier that for both low sizes and resolutions, the number of pixels available may be hardly sufficient for a decent representation of the character shape. For long, there was no known technology to render readable characters automatically from outlines – let alone to do justice to character design elements. Therefore, hand tuned character bitmaps were the only way.

Today's intelligent font scaling algorithms rely on additional instructions – appropriately called "hints" – that are defined and stored with the character outline descriptions. These will direct the character rendering software to uniformly represent stem widths (regardless of rounding effects), serif shapes (regardless of grid fitting), etc. (Figure 2). Hints cannot create extra pixels on the screen, and therefore cannot really improve the precision of the character bitmaps. However, they can improve the look of the text by dealing with the approximations in a consistent way.

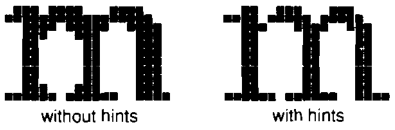

Figure 2. Effects of scaling hints.

The PostScript environment distinguishes between two types of outline fonts. The so called "Type III" fonts are user defined, and consist of normal PostScript instructions in clear text form. They contain no hint information, and can therefore only be used with good quality results on high resolution devices, or at larger point sizes (as for headlines).

The "real PostScript" or "Type I" fonts use special PostScript commands in the outline description, and are encrypted for even greater compactness. Type I fonts have embedded scaling hints, so that they will provide quality output at any size or resolution. They are currently available only from Adobe Systems or from type vendors that have licensed Adobe's tools. The recent publication by Adobe of all Type 1 font details will result in a wider offering of fonts encoded in this format, as well as in tools for end users to make their own hinted fonts.

Adobe's automatic font scaling system is based on the concept of having an intelligent intelligent rasterizer processing relatively simple, well-behaving outlines. As an example, the method expects a mathematically identical definition of all serifs that should render identically at any size. That concept of "outline regularization" became quite controversial among typographers, fearing that such artificial restrictions removed the elegance and the refinement of the original "true cut" design.

It is true that the very first PostScript typefaces contained outline changes to work around problem situations with early RIP software. In many cases though, the corrective actions were nothing more than removing errors from the outline databases, originally compiled by optical scanning or manual digitizing lettercards. Again, the rasterizer implementations have improved, and the font designers now better master the font encoding and hinting techniques, so that the necessity of regularization – and thereby its use – are significantly lower.

A full PostScript system can use a single font for all purposes (Figure 3). This supposes the use of a PostScript RIP with both the plain paper proofer and the high resolution imagesetter. In addition, the workstation relies on Display PostScript technology to drive the screen.

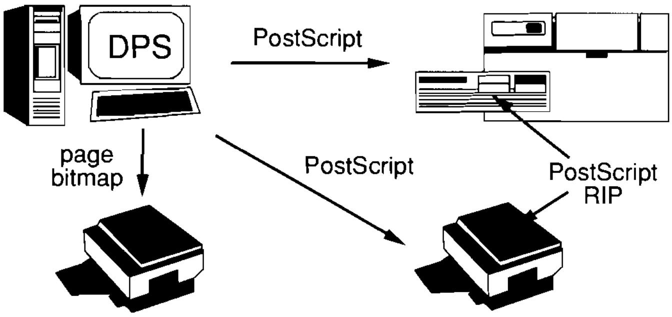

Figure 3. Full PostScript system.

Display PostScript however is not (yet?) widely available on PCs and workstations. Most application environments still rely on their own imaging model, or on the one generally supported by the PC or workstation itself. It is far more easy to match or duplicate the strictly "graphics" portion of PostScript than it is for the text side. The Adobe Type Manager (ATM) product was developed precisely for such situations. ATM packs the PostScript character rasterization software of into a separate software module, to be used as an add-on to the standard display/windowing system of the platform. Designed specifically to improve the rendering of text on low resolution displays, ATM exclusively uses the hinted Type I fonts (Figure 4). If the display software, in addition, can prepare a higher resolution bitmap to directly drive a dumb raster output device, then ATM will generate better character bitmaps for such output as well.

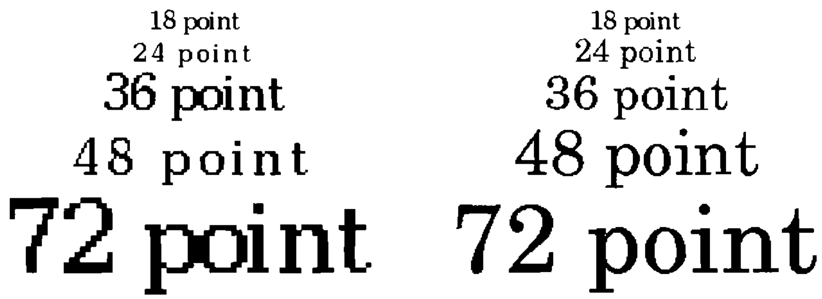

Figure 4. Comaparison of screen text quality when scaling bitmaps (left) or using outlines (right).

ATM, in fact, contains a more sophisticated rasterizer than the typical PostScript RIP available today. It provides better readability at small character sizes, especially in bold and italic variations, and requires less regularization, even less hints. Adobe has announced that the ATM technology will eventually become a part of the next generation PostScript RIPs.

The Apple LaserWriter was the first commercially available output device to incorporate a PostScript RIP, and desktop publishing on Apple Macintosh computers has been the major promoter of PostScript output. Some time ago, however, Apple decided to develop its own font scaling and rendering technology to complement the QuickDraw graphics environment. The Apple approach, originally known as "Royal", has since been endorsed by Microsoft as well, and will appear on the market later this year under the name of "TrueType".

There are some interesting differences between the two approaches. Adobe's Type I format uses third order Bézier curves in its encoding, which result in very compact outline descriptions. Apple's TrueType rather uses second order quadratic functions, that can be calculated more rapidly.

Adobe opts for relatively simple outline data, and puts most intelligence into the rasterizer itself. The benefit thereof is that existing fonts always can take advantage of better rendering software, as this becomes available. The Adobe option also makes font development somewhat simpler (although still far from automatic or straightforward). Apple bundles most of the scaling intelligence with the font outline description. The rasterizer itself becomes a relatively simple and compact piece of software. TrueType has a very elaborate hinting language, with typographic capabilities reaching beyond Adobe's – or most other – scaling methods. The font design, and hence the quality of the character representation, will be as good as the sophistication of the scaling hints included by the typeface producer; none other has control on the final result. It will undoubtedly take quite some time, training and expertise – and a clear market demand for the extra effort – before most type vendors will push the technology to its limits.

As of today, only Adobe licenses software tools required to produce Type I fonts with all bells and whistles. Most traditional type vendors having licensed these tools and have reported going through an extensive learning phase. The first typefaces are now passing their rigorous quality assurance process. The publication of the Type I specifications will certainly open other font sources. Apple will rely on third parties to provide TrueType production tools, and expects products from traditional type suppliers as well as desktop offerings targeted at the end user. The first typefaces and tools are expected to appear together with the release of Apple's System 7 operating system version for the Macintosh, later this year. Broad font offerings from classic type libraries will here too come only after the traditional vendors have gone through some painful, but unavoidable learning and debugging phases.

When Microsoft declared its support of TrueType, Apple announced that it would utilize the Bauer PostScript clone RIP (acquired earlier by Microsoft) with future output devices. The new joint technology goes by the name of "TrueImage", and – of course – incorporates TrueType for text rendering.

Neither Apple nor Microsoft plan to use (Display) PostScript technology for screen display purposes. As an example, a typical Macintosh application will continue to use the QuickDraw environment, whereby TrueType technology is used to display text on the screen. TrueType may also supply the character bitmaps for printing to dump raster devices, the remainder of the page bitmap built by other QuickDraw routines. For final output, the QuickDraw display list can be translated into PostScript-style commands and sent to a TrueType (i.e. PostScript clone) RIP. The RIP too will use TrueType outline fonts, that can be downloaded from the workstation (Figure 5).

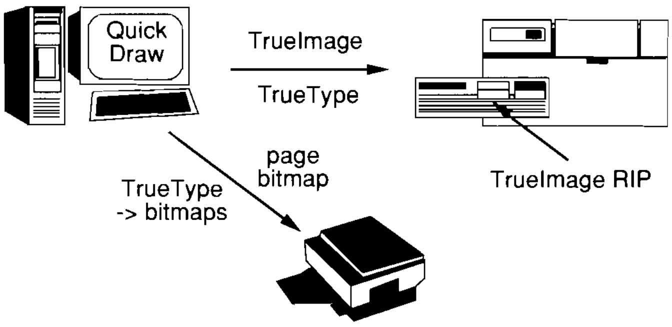

Figure 5. QuickDraw/TrueImage system.

In the same way, OS/2 and Presentation Manager applications will adhere to the GPI imaging model. In both cases, there are significant differences between the workstation's internal imaging model and the PostScript one used by the output RIP. Generating a TrueType command stream will require conversions that are not always simple or precise. Apple's and Microsoft's plan is to extend TrueType with additional operators, to support QuickDraw and GPI in a more direct way. It is unclear whether, in addition, TrueType will have some language restrictions compared to the original Adobe PostScript.

Apple's TrueType PostScript interpreter will also accept downloadable Type I fonts. In addition, Apple has announced that it is developing a method to download TrueType fonts to "real" Adobe PostScript RIPs such as the one in its own LaserWriter. Of course, once there is a significant market using primarily TrueType fonts, Adobe may choose to add direct support for TrueType into its products as well.

To most observers, the so called "font war" is fought between Adobe, Apple and Microsoft. IBM decided on the latest battle, by announcing support for Adobe Type I font technology across all systems under its Systems Applications Architecture (SAA). There are however two more players on the field, moving quietly but not less successfully.

A prominent player in the office and corporate markets is Hewlett-Packard. LaserJet printers and clones still outsell PostScript devices five to one. HP has recently introduced the LaserJet III printer series, whose PCL Level 5 command language adds scalable fonts as well as HPGL plotter compatibility. The font technology used with the printers is AGFA Compugraphic's "Intellifont", which is also available as separate software modules for screen display in DOS and OS/2 PC environments. For those users requiring even more graphic sophistication, HP offers a PostScript cartridge as an option to the printer RIP.

In the Unix workstation world, Sun Microsystems has acquired the Folio font technology and included it as "TypeScaler" into its X11/NeWS windowing system. It is worth noting that NeWS, by itself, is a close replica of the Display PostScript environment. The Sun implementation adds scalable type functionality to the XWindows environment as well (whereas the "standard" only covers bitmap fonts). TypeScaler is a very fast rasterizer, generating screen and printer fonts of fair quality on-the-fly. It is currently the only system with proven automatic translation into the Folio F3 outline format, including automatic hint generation. Sun workstations are widely used in corporate and professional publishing systems. The success of the Folio technology will further depend on the actual availability of compatible output devices.

With all these competing output and font technologies, the overall picture for the publishing industry becomes quite complicated. Figure 6 illustrates how the situation will look, in the near future, for some of the major PCs and workstations used or promoted for electronic publishing. In two cases – the NeXT computer and IBM's Series 6000 Unix workstations – a full PostScript implementation is available. Sun workstations have NeWS (very close to Display PostScript) plus TypeScaler at the display side, and PostScript at the output. System 7 for the Macintosh will use QuickDraw with TrueType for screen display and output on simple devices, and TrueImage for more complex ones. The OS/2 -Presentation Manager team will offer both ATM and PostScript, as well as TrueType and TrueImage, and even provides hooks for other additional solutions, such as Intellifont and PCL5.

However, add-on implementations of the Adobe Type Manager software are available for Macintosh as well as for Sun systems. In addition, there is the Mac LaserWriter driver supporting PostScript output. The combination of all of the above makes PostScript presently the only choice for cross-platform compatibility.

The question may arise whether the total market is large enough to accommodate so many different font standards. The use of typography is however no longer limited to traditional publishing. PC applications including word processing, presentation graphics and desktop

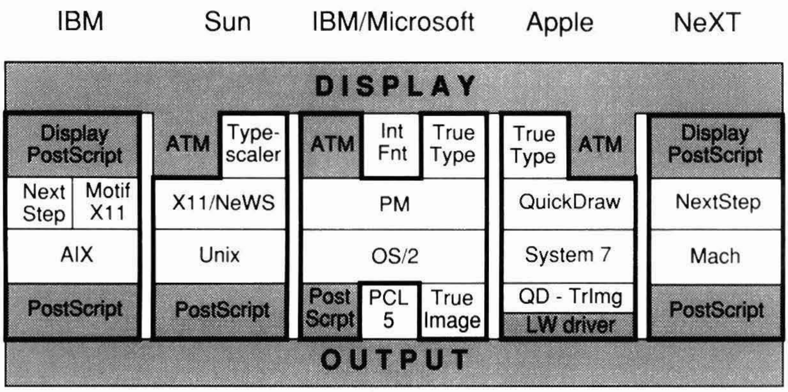

Figure 6. Overview of system architectures.

publishing increasingly provide support for type (Figure 7). In addition to the captive market (typefaces for the traditional, closed composition systems), there is a huge emerging open market reaching to the millions of PCs and workstations in today's offices and businesses. It is not inconceivable that multiple font standards each obtain a comfortable marketshare in specific areas. Accepting documents from these diverse environments – with their own "local" font standard – will be one of the challenges for the publishing professionals. At least for the moment, there are no good-enough font translators in sight. Multiple libraries will have to be used, with most probably support for multiple font formats by workstations and RIPs. It remains a concern how closely different encodings of the same typeface will match each other as for widths, shapes and character sets.

At the output side, there is a risk for multiple, incompatible evolution paths for page description languages, with now two major industry forces shaping the PostScript of the future. Each camp has different goals, different priorities, different markets to serve. International standardization efforts, such as ISO's Standard Page Description Language (SPDL) development, may perhaps bring some focus to the industry.

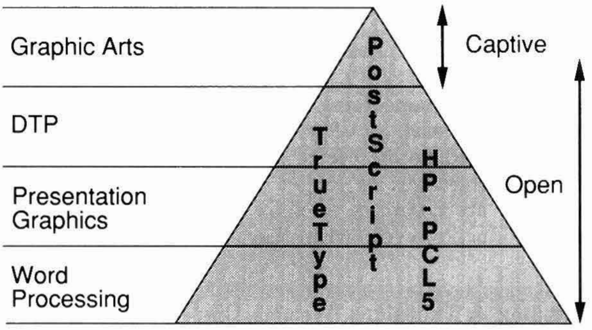

Figure 7. Type markets.