解　説　3. DTP の要素技術—技術動向の現状を知るために—

3.3 ページ記述言語の現状と動向†

石　田　晴　久††

1. はじめに—ページ記述言語とは何か？

ページ記述言語 (Page Description Language, 略して PDL) とは，文書の中の各ページにある情報の空間的（物理的）な構成や配置を出力装置（プリンタやディスプレイ）向けに記述するための一種のプログラム言語である．ただ言語とはいっても，人間のプログラマが直接使う言語というよりは，ワープロ・ソフトウェアやレイアウト・プログラムあるいは図形処理プログラムなどの応用ソフトウェアで自動的に生成されるデータの形式としてもっぱら使われるところに大きな特徴がある．

つまり，PDL で記述されるソース・プログラムは応用ソフトウェアからみればデータであり，出力装置側からみれば制御プログラムになっている．この関係を図-1 に示す．PDL の発想のユニークな点は，データ＝プログラムという図式が成り立っている点である．

他の多くのソフトウェア概念と同じく，この PDL の概念が生まれたのはアメリカである．PDL は当初，レーザプリンタに対する単なる制御コマンド群として開発されたが，そのうちにマクロからさらに進んで手続きが導入され，低レベルではあるが，プログラム言語の性格をもつに至っている．その過程では，1976 年に Evans & Sutherland コンピュータ社で開発された Design System (J. Gaffney らによる)[4]，1978 年にゼロックスで開発された JaM (M. Newell と J. Warnock による)[1] およびやはり同社で 1982 年ごろに開発された Press[1] などが重要な役割を演じた．

今日比較的広く使われている PDL としては次の三つがある．

PostScript　Adobe システムズ社で 1984 年に開発[4),5)]

Interpress　ゼロックス社で 1983 年ごろに開発[2),3)]

DDL　Imagen 社で開発[1]．HP (ヒューレット・パッカード社) の PCL はこれと互換性をもつ．

このうち最もよく知られ広く使われているのは PostScript である．これについては次章以降でやや詳しく述べる．次の Interpress は 1984 年に発表された PDL で，PostScript のお手本になっていることから想像できるように，両者は非常によく似ている面がある．PostScript に比べた Interpress の利点は，(ページごとのみならず) 文書全体の制御がより容易，プログラムが基本的に binary のため (読みにくいが) 高速，ネットワーク上での分散処理向き，ライセンス料なしで使用可能といった点である．

一方，DDL について特筆すべき点は，前 2 者と似ていることと，HP 社の普及型レーザジェット・プリンタでも使われていることである．

さて，我が国ではというと，レーザプリンタや LED プリンタなどのページ・プリンタ用にプリンタ制御言語は開発されて使われているが，それらは各社マチマチであり，我が国独自の本格的な PDL は開発されていない．外資系を除くと，数社が Adobe 社で日本語化された PostScript を採用しているのが現状である．

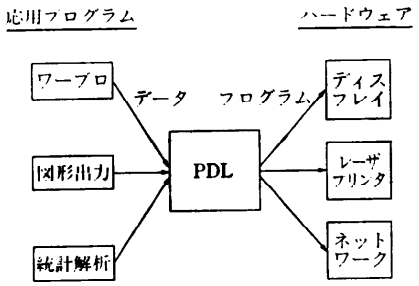


図-1　応用プログラムをハードウェアから独立させる PDL (ページ記述言語) の位置

† State-of-the-Art of and Trends in Page Description Languages by Haruhisa ISHIDA (Computer Centre, University of Tokyo).
†† 東京大学大型計算機センター

2. PostScript の機能

PostScript は，ゼロックス・パルアルト研究所を去って，1982年に Adobe 社を設立した J. Warnock らが1984年に発表した PDL である．この PDL は，1985年にアップル社が発売したレーザライタ（レーザプリンタ部はキヤノン製）にそのインタプリタが 1 MB の ROM の形で内蔵されるようになってから世に知られるようになった．このレーザライタは同社のパソコンであるマッキントッシュや UNIX ベースの他社のワークステーションで広く使われている．これには PageMaker などのレイアウトソフトウェアで PostScript がサポートされ，UNIX の TeX, troff, plot（図形出力），S（統計解析）などのソフトウェアの出力を PostScript に変換するプログラムやそのための機能が普及したことによる．

さて，ここでは紙数の都合で，PostScript 言語の詳細には立ち入れないから，簡単なプログラム例を示すことにする．**図-2** は，グレイ・スケールの応用例である．PostScript では，白黒やカラーの情報以外に，白黒の濃さを黒 (0.0) から白 (1.0) の間の値で指定することが可能である．そこでこの例では，for ループの中で，setgray でセットすべきグレイ・スケールを 0（黒）から始めて 0.02 きざみで少しずつ白くし，0.98（ほとんど白）まで変えている．そしてその間に，x 方向左側へ 1 ポイント，y 方向上へ 0.5 ポイントずつ原点を移しながら PDL という文字列を表示させた．最後は 0 setgray により黒で PDL を書かせている．このやり方では出力に時間はかかるが，PostScript ならではの出力がえられて面白い．

```
% Show gray-scales
/Times-Italic findfont 80 scalefont setfont
/printPDL {0 0 moveto (PDL) show} def
320 400 translate
0 .02 .98
 {setgray printPDL −1 0.5 translate} for
0 setgray printPDL        % draw in black
showpage
```

（a） プログラム例

（b） 出力（これをうるには上記のソースの先頭に %! という行を付加する必要がある）

図-2 PostScript プログラムの例とその出力

この PostScript は，図形記述言語としての性格ももっている．スタックを使う逆ポーランド的な記法のため，“if（条件）実行文”の形の構文が，“条件 {実行文} if”の形になる点などは慣れないと分かりにくいかもしれないが，慣れた人にとっては，PostScript は UNIX の plot 並みの記述力をもつといってよい．人によっては，グラフィック言語の国際標準である GKS よりも PostScript のほうが使いやすいという見方をする人もある．

PostScript でもう一つ工夫されているのは，写真やファックスなどの画像の扱いである．これらは，1画素当たり n ビットにディジタイズした後に，数値データの配列の形にしておけば，拡大・縮小・移動・回転などが自由にできる．

次にこの PostScript の特長をあげてみる．

（1） 一部のパソコンで使われている FORTH に似たスタック指向で比較的低レベルなプログラム言語である．

（2） 文字テキストのほかに，図形やスキャンされた画像（モノクロ，グレースケール，カラー）も統一的なイメージ・モデル（GKS や PHIGS に対抗）で扱える．

（3） オペレータ（手続き）が数多く定義されていて，文字サイズの変更や図形の回転など，豊富な処理が表現できる．図-2 の findfont, scalefont, setfont, moveto, show, def, translate, setgray, for ループ，showpage などは，全部で 274 個ある基本オペレータの例である．

（4） データの長さはポイント（1/72 インチ）などの単位で表されるため，出力装置のハードウェアの解像度やサイズなどには左右されない．図-1 に示したように，いろいろな装置に対してデータの標準化ができ，装置からの独立性が保てる．

（5） 英字については 5000 種以上といわれるほど多種多様な輪郭線フォント（拡大縮小自由自在な）が用意されている．

（6） その言語仕様が参考文献 4），5）にあげた本の形で公表されている．PostScript プログラムはレーザライタなどですぐテストや実行ができる．Interpress や DDL に比べると PostScript はずっとオープンである．

また構文も簡単で，たとえば，

① x y moveto
② α β γ {…} for
③ /v {サブルーチンの内容} def

のような形をとる．これらを実行する場合，①では x と y がまずスタックに積まれ，moveto でそれらを使って座標 (x, y) へのカレント・ポイントへの移動が行われる．②は，α を初期値，β をインクレメント値，γ を最終値とする繰り返しループである．一方③は v という名前のサブルーチンの定義で，こうして定義されたサブルーチンは単に v と書くだけで実行される．逆ポーランド記法だから，“(” と “)” のカッコは一切不要である．その代わり “(” と “)” は文字列を表すのに使われる．プログラムの先頭や末端を表す記号はとくにない．

3. PostScript の日本語化

前述のように，我が国では PDL の研究はそもそもあまり行われていないが，1986年の時点では皆無であった．東大大型計算機センターで，当時，分散型のマルチメディア文書処理を研究していた筆者のグループは，偶然 PostScript に着目し，1986年秋から1987年にかけて，PostScript のインタプリタを開発した．

これは274個のオペレータのうちの241個をほぼ仕様どおりにC言語で実現した英語用のものである．そのソース・コードは，コメントを除いて，約10,000行 (200 KB) となった．コンパイル後のロード・モジュールは，約 110 KB あり，それにいくつかのレーザ・プリンタやディスプレイに対するデバイス・ドライバを付加して使う形になっていた．

そこでこのソースを 1987 年初めに PDS (Public Domain ソフトウェア) として公開したところ，すべてのパソコン雑誌に取り上げられるなど，大きな反響を呼んだ．しかしこれはあくまで英語対応の試作版でしかない．

我が国で PostScript を卓上出版の目的に使うとなると，当然必要となるのは，PostScript での日本語の扱いを可能にすることである．先の東大版のインタプリタでは実質7ビットの ASCII コードのみならず，8ビット・コードのデータも扱えるようになっていたから漢字の扱いも不可能ではなかったが，漢字仮名混じり文を本格的に扱うとなると，いくつかの拡張機能が必要となる．そこで筆者は私的に PostScript 研究会を開いて，関係メーカやソフトウェアハウスの人々とこの問題について検討を行い，同年9月に日本語化標準仕様第1次案をまとめ，これを Adobe 社にも提示した．これに対し，同社からは 1988 年3月にわれわれの提案をある程度盛り込んだ形の仕様が発表された．これはわれわれの案とは異なっているが，狙いは同じであるため，PostScript 研究会としても，標準化の趣旨からして，これに付帯条項をつけたうえで，第2次案として受け入れることにした．

これは，その後，PostScript の日本語機能の事実上の標準になっている．その骨子は次のとおりである．

（1） 複合フォント

従来は一つの（基本）フォントとして8ビットで表せる256文字の文字セットしか許されなかったが，Encoding の要素として，単なる文字のみならず，他の256種のフォントが指せるようにする．これが複合フォントである．

（2） マッピング

2バイト（または1バイト）の文字コードと複合フォントとの対応づけは，FMapType の値で区別する．ASCII コードは FMapType=1 だが，これが2のときは，上位バイトで（JIS コードの区に対する）基本フォントを，下位バイトでその中の1文字を選択する．たとえば，亜唖娃の JIS コードは 16 進でそれぞれ 1601, 1602, 1603 になっているが，この場合上位の16が区を表している．

（3） タテ書きのメトリックス（寸法指定）

従来は，**図-3**（**a**）の横書き用の寸法しか指定できなかったが，**同図**（**b**）のように縦書き用の寸法（w1 と v）も指定できるようにする．横書き・縦書きの区別をするパラメータは WMode である．

（4） 外字の扱い

複合フォントの空いているところへ入れる．

（5） フォント辞書項目のキーの新設と変更

EscChar（エスケープ・コード），WMode，PrefEnc，Metrics2，（縦書き用情報），CDevProc が追加され，FontType（基本フォントと複合フォントの区別）および Encoding の機能が拡張された．これらはひと口で説明するのは困難だから，詳しくは文献5）をみていただきたい．

（6） オペレータの新設と変更

紙数の関係で詳述しないが，setcachedevice2（縦書き用），cshow（文字列出力用），rootfont（複合フォント用）の三つが新設された．機能が拡張されたオペレータとしては widthshow，awidthshow，kshow，currentfont，setcachedevice などがある．

（7） プログラムとデータ

コメントや文字列の中に漢字かなまじり文が書ける．（変数名などの名前として日本語名をつけてもよいが，プログラムの国際的な互換性を保つうえからはすすめない．）

以上の Adobe 仕様に対して，われわれの日本語 PostScript 研究会で付け加えたのは次の2点である．

（8） 日本語の内部コード

各8ビットの2バイトの EUC (Extended UNIX Code) を使用する．筆者が個人的にほしいと思うのは，各7ビットの2バイトからなる JIS コードのサポートである．これはネットワークに PostScript プリンタを直接つなぐときなどに必要となる．

（9） 日本語フォントの標準名

Mincho および Gothic を共通のフォント名として用いる．

PostScript で日本語を扱ううえで，さらに重要なのは漢字のフォント，それも輪郭線フォントを用意することである．もちろん，ドットの集まりで表現されるフォントでも，24×24 ドット程度のものでなく，64×64ドット程度になればかなり美しい字になるが，ドット方式のフォントでは，フォント・メモリの容量が過大になるうえ，拡大縮小したときにえられる字の形は必ずしも美しいものにはならない．

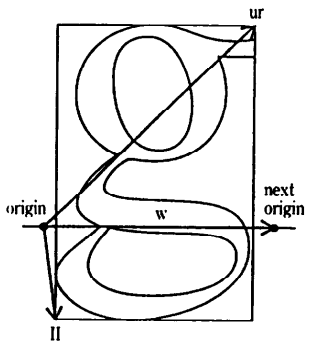
この点，PostScript が特長としている輪郭線フォントを使えば，原理的には拡大縮小をしても形はくずれない．ただ線の太さや間隔のバランスからいうと，一つの字体を極端に大きくしたり小さくしたりするとどうしても無理が生ずるから，うんと小さい字にはドット式フォント，それより大きな字用には，サイズによっていくつかの輪郭線フォントというように，キメの細かな対応が必要となる．

そうした輪郭線フォントは明らかに高価である．したがってフォントはなるべく多くの機種で共用できることが望ましいが，共用上障害となるのは，フォント・データには圧縮が必要だという事実である．この圧縮法は今のところ各フォント・メーカごとにマチマチとなっている．このため，各種プリンタ向けにフォント・データを標準的なファイル形式で提供することは現在不可能であり，PostScript の今後の問題点の一つとなっている．

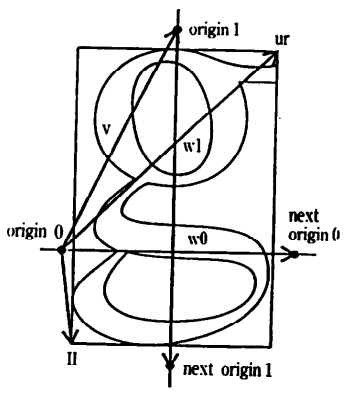
ところで，フォント・データ・ファイルの大きさは，フォントのデザインや種類にもよるが，漢字フォントを数種揃えるとして，各フォントごとにサイズにより3種のフォントを用意するとなると，30 MB 程度になることもある．日本語対応の PostScript プリンタはアップル社や NEC から発売されているが，いずれにも 40 MB の固定ディスクがついている．これに入っている日本語フォントは竜明体（明朝）とゴシック体の横書き体および縦書き体の計4種である．

4. PostScript のマルチウィンドウへの拡張

日本語への対応と並んで，PostScript で今後必要になる拡張はマルチウィンドウへの対応である．すでに，WS（ワークステーション）のメーカであるアメリカのサンマイクロシステムズ社が NeWS や Open Windows と呼ばれるウィンドウ・システムを WS に，また PostScript の開発元である Adobe 社が開発した Display PostScript は，NeXT 社の WS に標準装備され，DEC 社の DECwindows（Xウィン



（a） 従来のメトリック



（b） 拡張されたメトリック

図-3 横書きと縦書きの寸法指定（キヤノン提供）

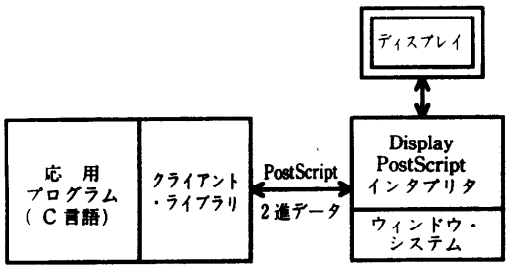


図-4 Display PostScript の使われ方

ドウ対応）にも採用されている．

マルチメディア・データをプリンタに出力する場合は，ただ出力すればそれでよいが，同じデータを画面に出すとなると，画面のうえでのスクロールやマルチウィンドウの切り換え，またキーボードやマウスを通したユーザとのやりとりも必要となる．WS がマルチタスクで動く場合に備えて，共用メモリを使ったマルチタスクの制御も入れなければならない．さらに Display PostScript では，図-4 のように使うクライアント・ライブラリが用意され，C言語を通して PostScript の機能が使えるようになっている．

一方 NeWS では，PostScript の 274 個のオペレータに対し，95個ものオペレータの追加が行われた．不幸なことに，これは Display PostScript の仕様とは異なっている．

問題といえば，PostScript インタプリタの高速化も将来の課題である．この言語で文字や図形や画像の回転を行うとき，sine や cosine の計算が必要になるが，それをソフトウェアのみで行うとすると，時間がかかる．（将来はこの処理専用の LSI が必要であろう．）

こんな状況だから，PostScript は現在のパソコンでは重すぎて使えない．そこで Adobe 社では，PostScript からグラフィックスやイメージ処理の機能を削除した文字処理専用の ATM (Adobe Type Manager) を開発している．これを使えばマッキントッシュや IBM PC で PostScript のと同じ輪郭線フォントが使えるようになるのである．

5. ページ記述言語の標準化

ここまで PostScript を中心に PDL について述べてきたが，メーカが PDL，とくに PostScript を採用するときの問題点の一つは，その言語仕様について，Adobe 社とライセンス契約を交わさなければならないという事実である．そこで，これを避けるためと，もっと一般的な互換性の向上という観点から，PDL の標準化の試みは当然行われている．

これに関して，ISO では，ODA (Office Document Architecture) の一環として，文書の論理構造（文書の型，章・節立てなど）を記述する SGML (Standard Generalized Markup Language) と並んで，文書の空間構造を記述する SPDL (Standard PDL) の検討が SC 18/WG 8 で行われている．この SPDL は PostScript をベースにしていて，オペレータはほぼ共通である．主な違いは，文字コードや文字フォントを限定していないことで，ISO のフォント標準化案 (DIS-9541) に準拠したフォントならなんでも使えるように考えられている．しかし SPDL が固まるまでにはまだ数年かかることであろう．

しかも現在の SPDL が固まったとしても，まだ残る問題としては次のものが考えられる．

（1） SPDL の国際版の規定．日本語 PostScript 仕様はすでにあるが，日本語のみならず，中国語や韓国語などを含む諸外国語について検討するのは容易であるまい．

（2） ディスプレイ版への拡張．Display PostScript をベースにすることが考えられるが，まだ流動的であろう．

（3） 文字フォントの規定．今後フォントの多様化とその圧縮技術の進展があるから，それをどう取り入れていくかは大問題である．

6. おわりに

PDL が普及すると，ユーザやソフトウェア・メーカさらにはハードウェア・メーカには次のような利点が出てくる．

（1） 応用ソフトウェアの出力が一定の形式になるからどのデバイスへでも出せる．

（2） デバイスは一定形式のデータのみ入力できればよいから，設計が楽で，標準化が実現する．

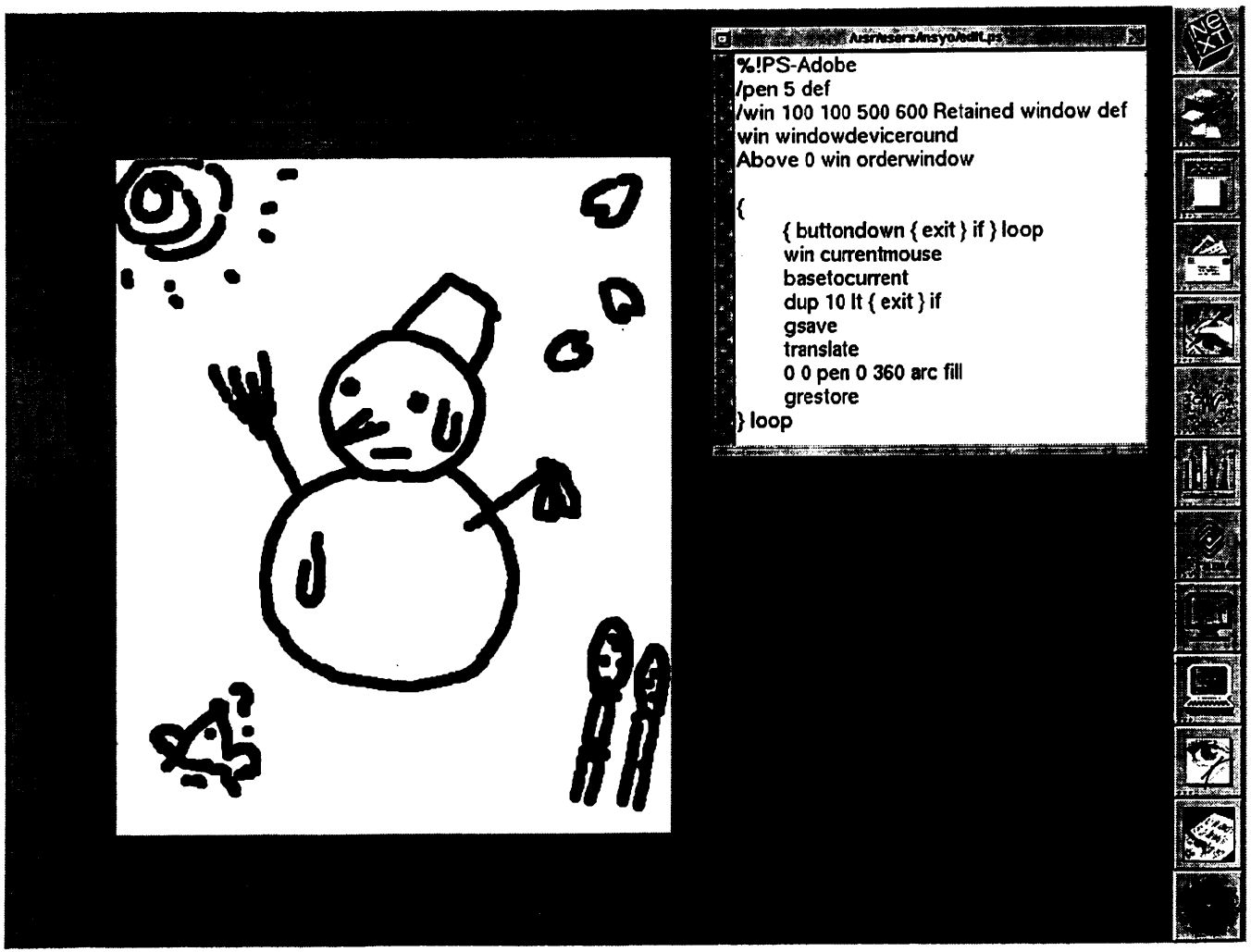
（3） 文字や図形の拡大縮小が自由になる．

（4） WYSWYG (What you see is what you get) 方式が実現しやすくなる．

（5） プリンタやディスプレイのソフトウェアからみた制御方式が統一され，互換性が高くなり，ソフトウェアが作りやすくなる．

しかし，PDL を普及させるには，少なくとも次のことが必要である．

（1） 日本語向きの PDL の標準化．我が国では PostScript では標準化ができない可能性があるから，



Display PostScript の画面例 (NeXT の場合)

もっと日本向きのPDLの研究開発をする必要がある.

(2) PDL向きのハードウェア技術の開発. 高速化のためには専用のLSIがいる. プリンタのメモリは固定ディスクではぐあいが悪い. 数MBのRAMと漢字フォント用に数10〜100 MBのROMが使えるようになる必要がある.

(3) 漢字の輪郭線フォントの多様化とデータ圧縮率の向上および低価格化.

(4) PDLプリンタの低価格化. 現在の日本語PostScriptプリンタは100万円前後する. これはパソコン・ユーザからみるとまだ高い.

(5) PDL形式のデータを出力する応用ソフトウェアの増加. これにはもちろんPDLプリンタがそもそも普及することが前提となる.

多くのパソコン・ユーザが嘆いているように, 現在のとくにパソコンのプリンタは各メーカの各機種ごとに制御方式がバラバラで, 標準化が行われているとはいい難い状況にある. このため, たとえば, 日本語ワープロ・ソフトウェアの開発元では, 何10種あるいは何100種類ものプリンタ用デバイス・ドライバの開発を余儀なくされている. こんなバカげたことに労力をさかれているようでは, プリンタや応用ソフトウェアの低価格化ははかれないし, いわんや卓上出版の発展は望めない. 今後は, 腰をすえてPDLの本格的な研究開発を行い, プリンタ・ディスプレイ・応用ソフトウェアの標準化をはかるべきである.

〔追記〕 Display PostScriptについては, 本稿を書いてしまった後に, Adobe社からThe Display PostScript System (1990) という部厚いマニュアルを入手した. これには, ディスプレイ用に追加されたPostScriptオペレータや図-4に示したクライアント・ライブラリ用のC言語関数 (ほぼ個々のPostScriptオペレータに対応) が, すべて説明されている.

本文で述べた274種のオペレータに追加されたオペレータは, 数えてみると101個ある. それらはウィンドウ系サポート (14個), 多重実行コンテクスト (11), ユーザパス (13), メモリ管理 (9), ハーフトーン定義 (8), 長方形 (8), ビュークリップ (6), フォント (5), ファイル・システム (6) などである. このほかカラー関係のcolorimage, setcolorscreenなど11個のオペレータも新設されている. したがって, PostScriptのオペレータの数は, 今や274+101+11=386ときわめて多くなった. このほか, 同じオペレータへの引数の追加なども行われているから, PostScriptはきわめて複雑なシステムになったといえる.

参考文献

1) Oakley, A. L. and Norris, A. C.: Page Description Languages: Development, Implementation and Standardization, Electronic Publishing, Vol. 1, No. 2, pp. 79-96 (Sep. 1988).
2) Bhushan, A. and Plass, M.: The Interpress Page and Document Description Language, Computer, Vol. 19, No. 6, pp. 72-77 (1986).
3) Harrington, S. J. and Buckley, R. R. (富士ゼロックス訳): インタプレス, 丸善 (1989).
4) Adobe (野中訳): ページ記述言語 PostScript チュートリアル & クックブック, アスキー (1989).
5) Adobe (石田監訳): PostScript リファレンス・マニュアル, アスキー (1988).
6) 石田晴久, 本岡茂哲, 杉山武信: PostScript 事始め, ①日経バイト, pp. 175-188 (1987年4月号), ② pp. 211-221 (5月号), ③ pp. 179-189 (6月号).
7) ページ記述言語 PDL 1 プログラム仕様書, イースト (1987).
8) D. A. ホルツガング (大木訳): ポストスクリプト入門, パーソナルメディア (1988).
9) 川端洋一: PostScript 日本語拡張仕様の概要, 日経バイト, No. 50, pp. 231-239 (1988).
10) SUN Microsystems: NeWS preliminary Technical Overview (Oct. 1986).
11) ISO: Information Processing—Text and Office Systems—Standard Page Description Language (SPDL), SC 18/WG8N-715 Rev (1989).

(平成2年5月31日受付)