特輯 技術解說

Page Description Language 技術

金 孝 男, 金 榮 柱
(주)큐닉스 컴퓨터 應用시스템硏究所

Ⅰ. 서 론

우리가 출력장치-조그마한 dot 프린터에서 대용량 컴퓨터의 규모인 imagesetter에 이르기 까지-를 통하여 출력물을 얻기 위해서는, 출력장치를 제어하기 위한 수단이 필요하다. 출력 장치를 제어하기 위한 수단, 즉 프린터 명령어를 넓은 의미의 PDL(page description language)이라고 한다. 한편 출력장치의 특성상 글자단위, 줄단위, 또는 전체 페이지 단위로 출력할 내용을 처리할 필요가 있기 때문에, 프린터 명령어도 이러한 특성을 감안하여 고안되고 있는데, 이중에서 페이지 단위의 출력을 하기 위한 프린터 명령어를 좁은 의미의 PDL이라고 한다.

협의의 PDL은 고가의 laser printer나 typesetter, imagesetter 등에서는 상당히 오래전부터 사용되어 왔으나, 이들은 극히 일부의 사람들만이 사용하는 출력장치이므로 일반 대중의 인식도는 미미한 상태이었으며 1980년대에 들어와서 저가의 일반 업무용 laser printer가 보급되기 시작하면서 일반 대중에게도 알려지기 시작하였다.

본 고에서는 이러한 협의의 PDL에 대한 현재의 기술 수준 및 앞으로의 발전 방향에 대하여 기술하겠다. Ⅱ장에서는 현재 많이 사용되고 있는 PDL의 종류 및 기술수준에 대하여 기술하였으며, PDL의 표준으로 인정받고 있는 PostScript에 대하여 Ⅲ장에서 별도로 좀 더 자세하게 설명하였다. Ⅳ장에서는 PostScript이 프린터 명령어로만이 아니라 일반적인 imaging model로서 앞으로의 발전 방향에 대하여 설명하였다.

Ⅱ. PDL의 종류 및 현황

페이지 단위의 출력을 하는 출력장치에 사용되는 PDL은 각각의 출력장치를 공급하는 업체별로 그들 나름대로의 명령어를 정의하여 사용되어 왔으며, 이러한 형태의 출력장치는 상당한 고가의 제품이므로 사용자가 극히 제한되어 있고 사용자 측면에서도 그 의미를 이해할 필요가 없기 때문에 PDL에 대한 관심이 거의 없는 상태였었다. 그러나 1980년대 초 일반 업무용의 laser printer가 보급되면서 서서히 PDL에 대한 인지도가 높아지다가 1985년 Adobe Systems Inc.의 PostScript가 발표되면서 PDL이 일반 대중의 관심을 본격적으로 끌기 시작하였다.[1]

1970년대에는 프린터가 종이에 글자를 찍어내는 용도로만 사용되어 왔으며, 가상의 격자틀을 정의하여 그 격자틀에 정해진 글자를 찍는 용도가 전부이었다. 따라서 글자의 크기와 종류가 한정되어 있었으며, 한정된 글자를 격자틀에 찍어내는 것이 전부이었다. 이러한 형태의 printer중 대표적인 것이 Diablo 630이라는 daisy-wheel printer이다. Dot printer는 이보다는 좀 더 유연성을 갖고 있는 프린터이다. 출력 image를 조그마한 dot의 집합으로 처리하기 때문에 글자의 경우는 크기나 형태에 있어서 필요에 따라 여러가지 변형을 지원할 수 있으며, 간단한 graphic 기능도 지원할 수가 있다. 그러나, 기계적인 특성상 줄단위로 처리를 해야 하기 때문에 상당한 제한이 있으며 따라서 이러한 dot printer에 사용되는 프린터 명령어도 이러한 기계적인 특성이 고려되어 full graphic 기능을 지원하지는 않는다. 이러한 형태

의 printer중 대표적인 것이 Epson printer이다.

일반 업무용의 laser printer가 소개되면서 full graphic 기능을 갖고 있으며, 출력이 페이지 단위로 처리되는 진정한 의미의 PDL이 실현 가능해 졌다. 초기 단계의 laser printer의 프린터 명령어는 HP(Hewlett Packard)가 주도 하였으며, 몇가지 종류의 PDL-HP 용어로는 PCL(printer command language)-이 사용되었다. HP PCL은 처음에 PCL level 1이 발표된 이후에 계속적으로 개량되어 PCL level 4까지 발전되어 왔으나 PCL level 4까지는 그 당시까지의 하드웨어 및 소프트웨어 기술 수준 때문에 진정한 의미의 full graphic 기능을 지원하고 있지는 않다. Full graphic 기능을 지원하며, 글자꼴(font)의 형태도 자유자재로 변경시킬 수 있는 기능이 지원되는 진정한 의미의 PDL은 1985년 Adobe Systems Inc.이 발표한 PostScript로서, 이 PostScript가 PDL의 수준을 한 단계 높히는 데에 일익을 담당하였다.

PostScript의 출현으로 그 후에 나오는 PDL들은 PostScript에 버금가는 기능을 갖고 있으며, HP에서 PostScript에 대항하기 위해 발표한 PDL이 PCL level 5이다. PCL level 5와 PostScript는 강력한 full graphic 기능을 제공하며, 크기의 변경이 자유로운 외곽선 폰트를 지원한다는 점에서는 기능상 유사한 점이 있으나 기본적인 개념 및 language의 형태는 판이하게 다르다. PCL 5는 PCL 4와의 상향 호환성을 유지하면서 PostScript의 기능에 필적하는 기능을 보유할 수 있도록 PCL 4에서 발전된 것이며, 따라서 language의 형태도 흔히 보는 프린터 명령어 방식으로 되어 있다.[5] 반면에 PostScript는 FORTH와 유사한 한개의 programming language 형태로 되어 있으며, 이러한 language로 한페이지의 내용물을 기술하도록 되어 있다. 이러한 PostScript 언어로 표현된 내용이 프린터로 전달되면 이 언어에 대한 interpreter가 이 language의 내용을 판독하여 그 페이지의 내용을 생성하여 출력하게 된다. 즉 PCL 5의 경우는 프린터 명령어가 출력할 내용의 사이에 추가되어 매 단계에서 프린터를 어떻게 제어할 것인가를 지시하는 반면, PostScript의 경우는 한 페이지를 어떻게 구성할 것인가를 먼저 정의하고 그 페이지 내에 어떤 image를 출력할 것인가를 매개 변수로서 표시하고 있다.

PostScript가 주도한 PDL이 확산되면서 가장 변화가 많이 생긴 부분이 글자꼴(font)에 대한 처리 부분이다. 글자꼴이라는 것은 우리가 항상 대하는 것이기 때문에 그렇게 복잡한 것으로 생각되지 않으나, 실제적으로는 매우 어렵고 미묘한 분야이다. 우리가 항상 대하고 있기 때문에 그리고 사람의 눈이 어떤 면에서는 상당히 정교하기 때문에 글자꼴에 조그마한 불균형이 있어도 그것을 즉시 간파하기 때문이다. 또한 출력물의 내용을 마음대로 하기 위해서는 여러가지 형태와 크기의 글자꼴이 필요하게 된다. 이러한 문제를 해결하기 위하여 PostScript와 PCL 5에서는 글자꼴의 data를 외곽선 방식으로 가지고 있다. 즉, 글자꼴을 글자모양을 형성하는 제어점들과 그 제어점들간을 연결하는 다항식으로 표현되는 곡선 및 직선으로 표현하여, 글자의 변형을 자유롭게 할 수 있도록 하였으며, 글자의 모양으로 출력을 할 때에는 이 외곽선 정보로부터 rasterization이라는 과정을 거쳐서 글자를 출력할 수 있도록 하였다.

글자꼴을 외곽선 형태로 가지고 있으면서 처리하는 방식이 PostScript에서 시작된 것은 물론 아니며, 그 전에도 많은 출력장치가 이러한 외곽선 방식으로 글자 꼴을 처리해 왔었다. 외곽선 방식으로 글자꼴을 처리할 경우 생기는 문제는 글자를 구성하는 pixel의 수가 적을때에 나타난다. 고해상도의 출력장치에서는 우리가 흔히 보는 크기의 글자도 충분한 pixel 수로 구성되기 때문에 이러한 문제가 노출되지 않으며 따라서 고해상도의 출력장치에서는 상당히 오래전부터 글자꼴을 외곽선 방식으로 처리하였다. PostScript에서는 이러한 외곽선 글자꼴의 문제를 일반 업무용 프린터의 해상도(300dpi)에서 해결했다는 점이 기술적인 진보에 해당된다. PostScript에서는 hints라는 개념을 도입하여 소프트웨어 기술로서 이러한 문제점을 해결하였다. PostScript가 발표된 후에 PostScript에서 글자꼴을 처리하는 방식과 유사한 개념을 도입하여 외곽선 글자꼴을 처리하는 방법이 여러가지 발표 되었으며, Microsoft / Apple의 TrueType, SUN이 인수한 Folio의 F3, HP에서 채용한 Compugraphic의 Intellifont, Bitstream의 Speedo, 등이 이러한 것들에 속한다.[6]

현재의 PDL 기술은 PostScript를 개발한 Adobe Systems Inc.이 주도하고 있으며, 근래의 몇가지의 PDL 기술에 대한 도전으로 인하여 새로운 사양인 PostScript level Ⅱ를 발표하는 등, 당분간은 PostScript가 PDL 기술을 주도할 전망이다.[2,7,10]

Ⅲ. PostScript

1. PostScript 개요

PostScript는 강력한 graphic 기능을 갖고 있는 컴퓨터 프로그래밍 언어이다. 다른 프로그래밍 언어와 마찬가지로 operator라 불리는 명령어와 operands 또는 data object로 구성되어 있다. 이 언어는 이미지와 그래픽 그리고 text를 한 페이지에 자유롭게 표현할 수 있게 하며, 이 언어로된 프로그램은 composition system에서 printer로 전달되는 입력데이타이다. PostScript을 지원하는 출력장치의 구성은 "인터프리터"와 "폰트"로 이루워졌으며, 인터프리터는 대개 전용 프로세서에서 수행되거나, PC에 있는 특별한 보드에서 수행된다.

2. PostScript Interpreter

PostScript를 지원하는 application 프로그램이 원하는 document의 PostScript 페이지 description을 PostScript 프로그램의 형태로 만들어 내며 이것이 인터프리터로 전달되어 PostScript 프로그램이 수행된다. 인터프리터가 PostScript 프로그램을 수행한 결과는 프린터 내에 있는 page momery에 표현되며, 이 것은 다시 프린트 할 수 있는 output을 만들어 내게 된다. PostScript 인터프리터와 output device(printer)는 함께 묶여(bundle) 있으며, application에 의해 black box로 처리된다. Black box의 의미는 end user와 직접 상호작용이 없거나 적다는 의미를 내포하고 있다. 인터프리터는 output device를 직접 제어하는 전용프로세서로 구현되어 있으며 인터프리터의 동작 형태는 연속되는 'print job'들의 처리과정으로 원하는 output를 만들어 내게 된다. 프로그래머와 PostScript 인터프리터가 직접적으로 상호작용(interact)하는 특성때문에 PostScript 프로그램이 수행되는 형태가 다른 interactive programming language(BASIC or FORTH)와 아주 유사한 면이 있다. 인터프리터의 또 다른 장점은 프로그램을 개발하기 위해 요구되는 tool이나 editor가 없더라도 programming environment의 미비점을 보완해 주는 것이다. 즉, PostScript 프로그램을 작성할 경우 line 단위로 interactive하게 처리 되기 때문에 PostScript의 특성을 매 단계에서 피부로 직접 느낄 수 있다는 것이다. PostScript 언어의 커다란 특징중의 하나인 interactive mode는 PostScript 프로그램 개발환경에서는 아주 유용하게 사용할 수 있는 인터프리터의 장점이다.

3. 프로그래밍 언어로서의 PostScript

이 절에서는 PostScript의 일반 프로그래밍 언어로서의 특징들을 설명하겠다. PostScript 언어는 우리가 친숙하게 접하는 프로그래밍 언어의 특징과 요소들을 빌어왔으며, syntax는 FORTH 프로그래밍 언어에서 사용하는 postfix notation을 적용하고 있다. 그리고 특수문자의 수는 아주 적으며 오퍼레이터를 나타내는 이름은 미리 정의되어 있지 않다. 데이타는 number, string, array 등을 포함하며 데이타로써 프로그램을 처리하거나 언어의 수행상태를 조작하거나 제어하는 특징들을 갖고 있다. 이런 특징은 Lisp 프로그래밍 언어로 부터 파생되었다. PostScript 프로그램의 수행과정은 PostScript 객체라고 불러우는 요소들을 수행하는 과정인데, 그런 객체(object)들은 data 범주에 속하는 number, boolean, string, array와 수행되는 프로그램 범주에 속하는 name, operator, procedure로 크게 두가지로 나눌 수 있으며 데이타와 프로그램 사이의 구별될 만한 차이점은 없다.

인터프리터에 의해 수행되는 객체(object)들은 두가지 형태가 있다. 첫번째는 PostScript 메모리에 미리 저장되어 있는 객체들의 모임들이다. 이 객체들은 procedure object를 이용하여 procedure안에 정의되어 있으며 인터프리터는 순차적으로 object들을 수행한다. 두번째는 PostScript 언어의 syntax rule에 따라 scan 되는 character stream들에 의해 새로운 object들이 만들어 진다. 각 object들은 scan 되자마자 즉시 수행되어지며 각 character stream들은 file이나 communication channel을 통해 입력되어 진다. 그리고 PostScript memory에 미리 저장된 string object로 부터도 새로운 object를 만들 수도 있다. 인터프리터는 array의 수행과 character stream 입력과정 사이의 전후(back and forth) 연결을 제어할 수 있다. 예를 들어 character stream에 의해 만들어진 object가 array (procedure) object이면 character stream 입력을 잠시 멈추고 인터프리터는 array에 정의되어 있는 object들을 수행하기 시작하며 수행과정이 종료되었을 때 array object 이후 character stream 입력과정을 처리한다. 이런 모든 과정의 제어들을 완벽하게 기억하기 위해 execution stack을 사용한다. Execution stack의 설명은 stack에 대해 설명할 때 얘기하기로 한다. PostScript 프로그램의 수행과정을 올바르게 이해하려면 각 object의 특성과 의미를 정확하게 이해하고 있어야 한다.

1) Syntax

PostScript는 하나의 프로그램에 대해 구문론적인

요소를 갖고 있지 않은 점이 다른 일반 프로그래밍 언어와 다른 면이다. 다시 말해 PostScript은 전체 프로그램을 일정한 구문론에 맞게 작성하지 않고 프로그램을 형성하는 object 각각이 구문론 rule에 적합하게 작성된다는 점이다. 또 다른점은 PostScript 프로그램이 수행하기 전에 프로그램 전체가 읽혀지는 것이 아니고 character stream에 의해 각 object를 생성하고 각 object의 동작을 수행하는 형식으로 차례로 프로그램을 읽으면서 수행된다. Object를 생성하는 과정은 일련의 character stream이 읽혀지면서 syntax rule에 따라 scanner가 character들을 모아 token이라는 것을 만들어 낸다. 이 token이 하나 이상이 모여서 object를 형성한다. Object의 동작과 내부적 표현을 설명하기 앞서 object의 구문적 표현을 설명하기로 한다.

(1) Character set

PostScript의 표준 character set는 프린트 할 수 있는 ASCII character set의 subset과 space, tab, newline으로 구성된다. 그리고 PostScript은 그외의 character도 허용할 수 있다. 여기서 space, tap, newline은 white space라고 부르며 동일하게 처리되며 분리자(delimiter) 역할을 한다. Special character는 string, procedure, name literal, comment와 같은 구문론적인 요소들을 분리 시키는데 사용한다. 그리고 special character와 white space character를 제외하고는 regular character라고 부른다. Comment는 character '%'로 표시하며 string 내부에 있는 것은 제외한다. Character %와 newline 사이의 모든 character들을 comment로 인식한다.

(2) Number and Name

PostScript에서 사용하는 수는 signed integer, real, radix number들로 구성된다. Radix number는 base# number인 형태로 표현하며 base는 2에서 36까지의 decimal integer이고 number는 이 base에 있는 수(0~base-1)이며 10부터는 영문자('A'에서 'Z')를 순차적으로 사용한다. Name은 regular character들로 구성되면서 number로 해석할 수 없는 것들을 의미한다. 이것들은 executable name이라고 하며 '/' (slash) 다음에 나오는 character set들은 literal name이라고 한다.

(3) String

PostScript에서 string은 괄호'( )'로 분리하며, string 내부의 특별한 character들은 '( )'와 ' ' 이다. String 내부의 '\'는 특별한 목적을 위해 'escape'로 사용된다.

(4) Array and procedure

Character '['와']'는 array를 형성하기 위한 self-delimiter이며, '['와']'사이의 object를 포함하여 array object를 형성하며 결과를 만들어 낸다. 예를들면 [123/abc(xyz)]는 name object인 '['와']' 그리고 integer object '123', literal name object 'abc'와 string object 'xyz'등 다섯개의 token으로 구성된다. 수행될 때에는 '['와']'인 name object의 수행으로 결과를 만들게 된다. Procedure는 '{'와'}'으로 분리시키며 executable array라고도 한다. Syntax는 array와 유사하며 semantic은 다르다. Array는 array를 형성하는 요소들을 scanning하고 난 후에 결과를 만드는 것이 아니고, 각 요소들은 scanning하자마자 수행하면서 결과를 만들어낸다. 그러나 procedure는 scanning만 하는 것으로 scanning하면서 object를 만들어 내지만 수행은 하지 않는다. Procedure를 형성하는 각 object를 scanning하고 나면 새로운 각 object들을 procedure object에 포함시키면서 interpreter는 단지 data만으로 인식하고 수행은 차후에 invoke 될때 이루어 진다.

2) Data type과 object

PostScript 프로그램을 수행할 수 있는 모든 데이타는 object이며 이 object는 type, attribute(속성), value를 갖는다. 각 object의 type들은 각각의 특성을 갖고 있으며, 종류는 integer, operator, real, file, boolean, mark, array, null, string, save, name, fontID, dictionary로 나눌 수 있다. 또한 각 object들은 simple object와 composite object로 크게 두개로 나눌 수도 있으며, composite object는 array, dictionary, string 들을 포함하며 그외 object들은 simple object에 포함시킬 수 있다. Simple object와 composite object의 가장 큰 차이점은 object를 copy할 경우에 발견할 수 있다. Simple object는 object의 type, attribute, value들을 모두 PostScript 메모리 상의 한 장소에서 다른 장소로 transfer하는데 반하여 composite object는 value는 copy가 안되며 단지 원래 object와 copy object가 동일한 value를 공유하게 된다. C와 pascal과 같은 프로그래밍 언어에서의 pointer 사용과 유사하다.

Dictionary는 PostScript object와 관련해서 key와 value의 쌍으로 구성되어 있는 table을 말한다. Postscript 프로그램은 dictionary에 있는 key-value의 쌍을 삽입하거나 key를 찾아 value를 가져오는 여러가지 operation이 있다. Dictionary를 접근하기 위한 세가지 방법이 있는데 첫번째는 operand로 제공된 특

정 dictionary를 실행하기 위한 operator가 존재할 경우, 두번째는 묵시적으로 현재의 dictionary를 수행하기 위한 operator들이 있는 경우, 세번째는 수행하려는 operator인 name object를 만날때 dictionary를 접근할 수 있다. PostScript은 dictionary를 관리하기 위해 dictionary stack을 갖고 있으며, bulit-in dictionary로 postscript operator의 이름을 포함하고 있는 system dictionary와 PostScript 프로그래머들이 사용하기 위한 user dictionary가 있다. 그리고 이들 두 dictionary는 dictionary stack의 제일 밑에 위치한다. File은 일련의 character들을 읽고 쓸 수 있도록 하기 위한 object이다. Save object는 PostScript의 memory 상태에 대한 shapshot를 표현하며, save, restore operator에 의해 만들어지며 처리된다. FontID는 특별한 object이며 PostScript 폰트구조에 사용된다.

Object의 속성(attribute)은 각 object에 대해 여러개가 될 수 있으며 object가 수행될 때 object동작에 영향을 줄 수 있으며 data로서의 처리 과정에서는 object 동작에 아무런 영향을 미칠 수 없다. 그리고 object의 속성중에 literal과 executable 속성이 있는데 가장 큰 차이점은 interpreter가 object를 수행할 때 object가 literal이면 data로써 operand stack에 push 되지만 executable한 속성을 갖고 있으면 그 object는 excution stack에 push되어 수행하게 된다.

3) Stack과 virtual memory

PostScript 인터프리터는 PostScript 프로그램의 수행상태를 나타내주는 네개의 stack을 관리한다. 네개의 stack은 operand stack, dictionary stack, excution stack, graphics state stack으로 나누며 operand stack은 PostScript 오퍼레이터의 수행 결과에 의해 만들어진 object와 operand인 object들을 관리한다. Dictionary stack은 dictionary object들 만을 관리하며 인터프리터가 name object를 수행할 때 name을 key로 하여 dictionary를 찾는다. Execution stack은 interpreter에서 object의 수행과정을 기억하고 있으며 execution stack의 내용에 따라 PostScript 프로그램이 순차적으로 수행하게 된다. 위의 세가지 stack은 독립적이며 그들을 접근하기 위한 방법은 서로 다르다.

PostScript 프로그래밍에서의 virtual memory를 VM이라고 간략하게 말하여 VM은 composite object (array, dictionary, string)의 value들을 저장하기 위한 장소로 사용된다. Operand, dictionary, execution stack은 PostScript 프로그램이 관리하는 임시 기억장소를 사용하며 VM에 저장되지는 않는다. VM에 접근하는 operator는 save, restore이며, save는 VM상태의 snapshot를 저장하기 위한 operator이고 restore는 저장된 VM 상태를 복구하기 위한 operator이다. Restore는 save에 대응되는 composite object의 모든 변화를 undo하고 원래 상태를 유지하도록 한다. VM의 크기는 매우 크지만 제한된 크기이며 system마다 다를 수 있다. VM의 값을 없애거나 VM의 resource(자원)들을 free 시키기 위해서는 VM을 restore하는 방법밖에는 다른 방법이 없다.

4. PostScript의 Graphic 기능

PostScript 언어의 특징 중 하나인 그래픽 지원에 대해 몇가지 특성들로 나누어 설명하면 다음과 같다.

- Graphic state operators: Graphics state를 나타내는 parameter들을 처리하는 operator로서 graphics state는 그래픽 operator들이 수행하는데 있어 전제조건이 된다.
- Coordinate system and matrix operator : PostScript 프로그램에서 정의한 좌표 체계를 output device 좌표체계로 바꿔주는 CTM (current transformation matrix)를 처리하는 operator들이다.
- Path construction operators: 어떤 형태의 모형이나 선들을 만들때 graphic state의 한 element인 current path를 사용한다. 즉 current path에 new path를 추가하고 완성된 path를 close하는 과정을 통해 임의의 모형을 구성하게 된다. 여기에서도 CTM이 사용된다.
- Painting operator: 현재 경로에 의해 정의된 임의의 모형에 image와 color를 삽입하거나 경로를 따라 선들을 연결하면서 임의의 형태를 나타내게 한다.
- Device setup and output operator: PostScript 메모리와 physical output device사이의 관계를 연결하며, 한 page의 내용이 memory에 완전히 형성되면 output operator의 수행에 의해 output device로 전달되어 한 page의 description이 만들어지게 된다.

1) Graphic state parameter

PostScript 프로그램에서 그래픽에 관련된 operator들은 graphic stack에 의해 처리되며 임의의 형태를 만들어 내기 위한 매개체 역할을 한다. Graphic state는 stack에 의해 관리되며 PostScript 프로그램은 "gsave" operator를 통해 현재의 그래픽 state를 graphics state stack에서 보존하도록 되어 있다. 그리고 원하는 graphic state을 복구하기 위해서는 "gre-

store" operator를 수행하면 된다.

2) User space와 device space

PostScript 언어는 페이지 상의 각 점들의 위치를 설정하는데 사용하는 default coordinate system을 사용하고 있다. Postscript 환경에서 PostScript program과 output device의 좌표계는 다르다. Output device에서 적용하고 있는 좌표체계를 output space 하고 있는데 이 output device는 output device에 따라 x, y축 방향이 다르고 해상도(resolution)도 다를 수 있으므로 output space도 output device에 맞춰 바뀔 수가 있다. 그러나 PostScript 프로그램에서 사용하는 좌표체계인 user space는 고정되어 있으며 user space의 좌표체계는 PostScript 프로그램을 작성하는 사용자 측면에서 구현되어 있다.

좌표체계를 이해하기 위해 원점의 위치와 x축, y축 방향을 정확하게 인식해야 하며 더불어 각 축의 unit length를 알아야 한다. User space에서의 원점 위치는 output의 왼쪽 밑부분이며 x축의 방향은 오른쪽으로 증가하는 방향이며 y축의 방향은 밑에서 위로 증가하는 방향이며 unit length는 1/72 inch로 되어 있다. 이런 규정을 default user space의 좌표체계라고 한다. 여기서 1/72 inch는 printer 산업에서 글자의 크기를 나타내는 기본 단위로 사용하고 있는 1/72.27 inch를 사용하고 있다.

User space에서 device space로의 변환은 graphics state가 갖고 있는 parameter인 CTM(current transformation matrix)를 이용하여 바꿔준다. 여기서 PostScript이 사용하는 matrix는 2차원 배열인 3×3 matrix를 사용한다.

즉, $\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ tx & ty & 1 \end{pmatrix}$

인 형태의 matrix를 PostScript 언어에서는 [a b c d tx ty]의 형태인 array object로 표현하고 있다. 좌표점 (x, y)를 (x', y')로 바꾸기 위해 1차원 방정식을 이용해서 다음과 같이 (x', y')를 얻을 수 있다.

$$x' = ax + cy + tx$$
$$y' = bx + dy + ty$$

임의의 모형을 변환 시키기 위한 방법으로 PostScript에서는 translation, scaling, rotation시 다음과 같은 형태의 matrix를 사용한다.

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{pmatrix}$ $\begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$ $\begin{pmatrix} cosA & sinA & 0 \\ sinA & cosA & 0 \\ 0 & 0 & 1 \end{pmatrix}$

translation *scaling* *rotation*

PostScript 언어를 얼마나 이해하고 있느냐 하는 문제는 user space 좌표체계와 device space 좌표체계를 얼마나 정확하게 이해하고 있느냐에 달려 있다고 볼 수 있다.

3) Painting과 image

Painting operator는 메모리에 있는 모형들을 현재 page에 그려내기 위해 메모리를 scanning하면서 임의의 모형을 페이지에 나타내는 operator이다. 일반적인 painting operator는 "stroke," "fill"이 있으며 특별한 operator는 "image"이다.

PostScript 프로그램에 의해 경로가 만들어지고 그 경로 내부에 점들의 위치 결정을 위해 두가지 규칙 중 하나를 사용하고 있다. 첫번째 방법은 non-zero winding number 규칙이다. 이 방법은 특정한 점이 path 내부에 속하는 지를 결정하는 알고리즘으로 특정한 점에서 임의의 한 방향으로 직선을 그린다. 이 직선이 path segment와 교차하는 시점에서 path의 방향에 따라 1을 증가하거나 감소시켜 최종 결과 값에 따라 특정한 점이 path에 내부에 있는 지를 결정할 수 있다. 특정한 시점에서 0으로 출발하여 직선이 path와 교차할 때 path의 방향이 왼쪽에서 오른쪽이면 1을 증가시키고 오른쪽에서 왼쪽 방향이면 1을 감소시킨다. 결과 값이 0이면 path 외부에 있는 점이고 그렇지 않으면 내부에 있는 점이다. 두번째 방법인 even-odd rule은 특정한 점이 path 내부에 있는지를 결정하는 방법인데 특정한 점에서 임의의 한 방향으로 직선을 그린다. 직선과 path segment가 교차하는 수에 따라 특정점의 영역을 결정한다. 교차점의 수가 홀수이면 path 영역내부에 있으며, 짝수인 경우는 path 외부에 위치한다.

자연현상을 대략적으로 표현하거나, 이것들을 종합적으로 표출하기 위해 PostScript에서는 image라는 명령어를 사용한다. Sampled image라는 용어는 어떤 color를 표현하고 있는 각각 sample value들을 갖고 있는 장방형 2차원 array이다. PostScript image는 array의 행과 열의 순서에 의해 scanning하면서 생긴 grey-level 값에 의해 정의된다. Grey의 정도를 나타내기 위해 PostScript에서는 halftoning기법을 사용하고 있으며 grey를 만들기 위해 새로운 halfton pattern

(screen)를 만들어 내야 한다. 이는 screen operator에 의해 만들어지며 screen과 parameter들은 CTM에 영향을 받지 않고 단지 device space에서만 영향을 받는다. 하나의 screen은 deviec pixel array 위에 halftone cell들로 이루어진 장방향의 바둑판을 덮어 놓은 것으로 생각하면 된다. 각 pixel들은 바둑판의 한 cell에 포함되며, 한 cell은 여러 device pixel 들로 구성된다. 바둑판형의 screen은 frequency와 angle의 값에 따라 여러 형태로 구성할 수 있다. Frequency는 inch 당 cell의 수이며 angle은 device 좌표체계와 관련하여 바둑판 선들의 기울기를 의미한다. Grey level은 증가과정 즉, 흑에서 백으로 변하는 pixel의 변화를 의미하며, pixel의 변화는 PostScript 프로그램과 procedure에서 정의한 spot function을 이용한다.

4) Fonts(글자꼴)

PDL에서 중요한 것 중의 하나가 글자꼴이다. 글자꼴은 자유자재로 확대, 변형시키면서 원래의 모양을 유지하기에는 bitmap 형태의 글자꼴로는 불가능하기 때문에 외곽선(outline) 형태의 글자꼴을 사용한다. Font 기술에 대한 설명은 본 특집에서 별도로 취급하고 있기 때문에 여기에서는 설명은 하지 않겠다. 현재는 대부분의 font format이 공개되어 있으며 이러한 여러 종류의 글자꼴간의 변환 프로그램도 많이 나와 있다.

PostScript의 글자꼴 처리에서 장점중에 하나가 사용자 정의문자를 지원한다는 점이다. PostScript에서 내부적으로 사용하고 있는 font format을 type 1 이라고 하고[9], 사용자 정의 문자로 사용할 수 있게 만든 font format을 type 3 format이라 한다. Type 3 format은 원래 logo나 특별한 모양 등을 글자꼴로 정의하여 사용할 수 있도록 하기 위하여 만들어 놓은 것이나, 우리나라에서는 한글/한자가 PostScript 글자꼴에 정의되어 있지 않기 때문에 현재는 type 3 format을 사용하여 한글/한자를 PostScript printer에서 사용하고 있다.

Ⅳ. PostScript의 미래

PostScript를 공동 고안한 Adobe System의 John Warnock의 말에 의하면 PostScript를 개발하면서 2가지 점에 주안점을 두었다고 한다. 하나는 종이 한 장에 어떠한 내용-text, 그림, image, 사진-이 있더라도 그것을 표현할 수 있는 프로그래밍 언어를 만들겠다는 것이고, 다른 하나는 일반 사용자가 그 언어가 어떤 것이라는 것을 될 수 있으면 모르면서도 사용할 수 있는 컴퓨터와 출력장치 사이에 있는 보이지 않는 소프트웨어 layer를 만들자는 것이었다.

후자의 목적은 GUI(graphic user interface)와 device driver라는 방식에 의하여 어느정도 성취가 되었다. 즉 응용 프로그램과 device driver에서 PostScript에 관한 것을 모두 해결해 주기 때문에 일반 컴퓨터 사용자는 PostScript printer를 사용한다고 하더라도 PostScript가 어떤 것인지 전혀 알 필요가 없다. 이는 device driver가 응용프로그램에서 사용하고 있는 image model을 PostScript image model로 변환을 시켜주기 때문이다.

전자의 목적은 다른 말로 표현하면 모든 2차원적인 내용--그것이 컴퓨터를 이용한 출력물이건, 광고 도안이건, 사진이건, 심지어는 어떤 그림이건-을 원래의 것과 어느정도의 오차는 있겠지만 PostScript이라는 프로그래밍 언어로서 표현할 수 있도록 하겠다는 것이다. 이러한 맥락에서 PostScript는 출력을 위한 언어가 아니라 통신을 위한 언어라고도 볼 수 있다. 예를 들어 우리가 어떤 종이의 내용을 fax를 통하여 다른 곳으로 보낸다고 할 때 scanning 및 transmission에서 수반되는 오류 때문에 원래의 내용보다는 조금 달라진 형태의 것을 받을 수 밖에 없다. 그러나 이러한 fax 전송을 그 종이의 내용을 표현하고 있는 PostScript 언어로서 전송한다면, PostScript 언어는 ASCII text로 되어있기 때문에 우선 전송되는 data의 양이 scan된 image보다는 작고 또한 전송상의 오류도 방지할 수 있어서 정확한 내용을 받을 수 있다. 이러한 PostScript를 이용한 fax 전송은 이미 이러한 방식을 사용할 수 있는 기기를 개발 중에 있으며, 1~2년내에 이 방식을 사용한 제품이 나올 전망이다.

PostScript는 graphic 기능이 강조된 언어이므로, PostScript 자체내에서 설정한 image model이 있다. 이 image model이 현재는 printer 위주로 사용되고 있지만 디자인 자체가 일반적인 graphic 모형을 표현할 수 있도록 만들어졌기 때문에 printer에만 국한하여 사용할 필요는 전혀없다. 따라서 PostScript의 image model을 image model이 필요한 다른 device에 사용 못할 이유가 없다. 이미 NeXT 같은 system에서는 화면상에서도 PostScript의 image model을 사용하는 display PostScript를 채용하였으며, 이외의 몇개의 업체에서도 이미 display PostScript를 화면의

image model로 채용하고 있다. 현재 Mac과 MS-Windows에서는 image model로서 각각 QuickDraw와 GDI라는 그들 고유의 image model을 사용하고 있으며, 따라서 화면의 내용을 PostScript printer에 출력할 때는 image model간의 변환을 해야 하기 때문에 device driver가 할 일이 많아지고, 변환과정에서 약간의 오류가 발생할 수 있다. 장기적으로 본다면 GUI하에서는 화면의 image model과 printer의 image model을 동일한 것을 사용하는 것이 바람직한 방향이다.

Adobe System Inc.가 PostScript를 이용하여 추구하고 있는 또 다른 분야는 서로 다른 platform에서 작성된 PostScript 화일을 서로 교환하여 수정 및 출력할 수 있도록 하자는 것이다.[8] 이는 PostScript를 단순한 image model로서가 아니라, 문서의 표현 방식의 표준으로 하겠다는 것으로, 성공여부에 귀추가 주목된다.

V. 결 론

PostScript는 영어문화권에서는 대단한 성공을 한 PDL이다. 그러나 원래의 design 자체가 영어문화권을 대상으로 하였기 때문에 글자의 수가 많은 동양문화권에는 사용하는데 문제점이 많이 있다. 최근에 발표된 PostScript level II[10]는 이러한 문제점들을 해결하기 위하여 몇가지 부분이 확장되었다. 현재 일본의 경우는 PostScript가 영어문화권 만큼 호응을 얻고 있지는 못하며 우리나라의 경우는 이제 시작단계이므로 조금 더 지켜보아야 할 것이다. 단지 한가지 부언하고 싶은 것은 이러한 PDL은 표준화가 이루어져야 효과를 배가 시킬 수 있으므로 초기 단계에 한글 PostScript에 대한 표준화 작업이 진행되는 것이 모든 사람에게 득이 되는 일이다. 마지막으로 PostScript에 대하여 좀더 자세한 내용을 알고 싶은 독자를 위하여 PostScript manual이 아닌 학습용 도서 2가지를 추천하고자 한다.[3,4] [4]보다는 [3]이 좀더 자세한 부분까지 설명이 되어 있는 PostScript 학습용 도서이다.

參 考 文 獻

[1] Adobe Systems Inc., PostScript 참조 메뉴얼, 휴먼컴퓨터 역, 오롬 정보 처리, 서울, 1991.

[2] 여미라, 한윤섭, "PostScript의 구성과 활용," 전기학회지, 39권, pp. 60-65, 1990년 7월.

[3] Frank M. Braswell, *Inside PostScript,* System of Merritt & Peachpit Press, Berkeley, 1989.

[4] Stephen F. Roth (ed), *Read World PostScipt,* Addison Wesley, New York, 1988.

[5] Kent Quirk, "The Language of Lasers," BYTE IBM Special Edition, pp. 203-208, Fall 1989.

[6] Seybold Report on Desktop Publishing, Seybold Publication, vol. 3, no. 8, April 3, 1989.

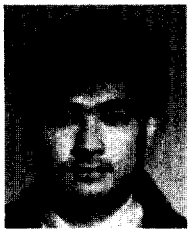
[7] Seybold Report on Desktop Publishing, Seybold Publication, vol. 5, no. 7, March 4, 1991.

[8] Seybold Report on Desktop Publishing, Seybold Publication, vol. 5, no. 11, July 22, 1991.

[9] Adobe Type 1 Font Format, Adobe Systems Inc., 1990.

[10] *PostScript Language Reference Manual* , 2nd Edition, Adobe Systems Inc., Addison Wesley, New York, 1990.

筆 者 紹 介



金 孝 男

1964年 10月 26日生
1988年 2月 홍익대학교 전자계산학과(학사)
1990年 2月 홍익대학교 대학원 전자계산학과(석사)

1990年 2月～현재 (주) 큐닉스 컴퓨터
주관심분야 : Laser Printer 관련 PDL 연구

金 榮 柱

1956年 8月 22日生
1979年 2月 서울대학교 자연대학 계산통계학과(학사)
1981年 2月 한국과학기술원 전산학과(석사)
1989年 2月 한국과학기술원 전산학과(박사)

1982年 3月～1985年 한국과학기술원 전산학과 TA / RA
1986年～1987年 한국과학기술원 전산학과 RA
1988年 5月～현재 (주)큐닉스 컴퓨터 응용시스템 연구소 책임연구원
주관심분야 : OS, Concurrent Programming Language, Communication.