PostScript의 구성과 활용-

여미라 • 한윤섭

(주)휴먼 컴퓨터

1. 서 론

Desktop Publishing(DTP)에서 원하는 페이지를 실제 출력기로 받아보기위해 필요한 가장 중요한 요소는 Page Description Language(PDL)이다. 현재 시판되는 응용 소프트웨어(S/W) - word processor, Computer Aided Design(CAD) 시스템, 혹은 DTP S/W - 는 무수히 많으며 새로운 S/W도 계속 쏟아져 나오고 있다. 또한 출력기 - laser, dot-matrix, ink-jet 프린터 - 도 이에 뒤지지 않을 정도로 다양하다. PDL은 이렇게 다양한 응용 S/W와 출력기 사이를 연결해 주는 다리 역할을 한다. 각 응용 S/W는 PDL형태로 결과를 생성하고 출력기는 생성된 PDL 프로그램을 수행하여 출력을 낸다. 그러므로 응용 S/W와 출력기는 서로 독립적으로 처리될 수 있다.

PostScript는 미국의 Adobe사에서 개발되었으며 1985년 처음으로 Apple의 LaserWriter에 탑재되어 시장에 등장하여, 현재 전세계 PDL 시장의 표준이 되어가고 있다. 그외에도 Imagen의 Document Description Language (DDL)와 Xerox의 Interpress등 여러 다른 PDL이 있다. 2절에서는 PostScript의 구성, 그리고 imaging model을 3절에서는 PostScript의 가장 큰 장점 중의 하나인 font를, 그리고 4절은 PostScript의 일반적인 활용 방법에 대해 서술하였다. PostScript에 대한 이해를 높이기위해 PostScript 프로그램의 출력 결과를 부록으로 첨가하였다.

2. PostScript Language

2.1. PostScipt의 구성

PostScript(PS)는 기존에 잘 알려진 C나 Fortran과 마찬가지로 원하는 작업을 컴퓨터가 이해할 수 있는 형태로 표현하기위한 프로그래밍 언어이다. 일반적인 프로그래밍 언어가 가지는 숫자(integer, real)나 배열(array), 그리고 스트링(string)과 같은 여러가지 데이타 형태와 조건문(if-then-else) 이나 loop(for, while)나 function 혹은 procedure와 같은 제어 구조를 가진다. 또한 일반적인 형태는 아니지만 정의되는 변수를 저장할 수 있는 key-value의 쌍으로된 dictionary라는 것도 있다. 그러므로 효율적이지는 않겠지만 PS로 일반적인 프로그램도 할 수 있다.

PDL이 일반 언어와 다른 점은 강력한 graphics 기능을 가진다는 것이다. PS에 정의되어 있는 graphics 관련 연산은 여러가지이다. 선이나 arc나 curve를 이용하여

임의의 모양도 만들고, 만들어진 모양의 윤곽을 그리거나 내부를 임의의 색으로 칠할 수도 있으며 주어진 모양을 clip에 이용하여 필요없는 부분을 제외시킬 수도 있다.

일반적인 text도 graphics와 완전하게 통합될 수 있다. PS에서는 text의 글자들도 graphics 형태로 처리하므로 graphics에 적용되는 모든 연산을 글자에도 적용시킬 수 있다. 즉 text나 graphics가 구분없이 어떤 형태로든 한 페이지 내에서 배치될 수 있다.

Sampled image가 사진과 같이 이미 존재하는 것으로 부터 만들어지기도 하고, 임의의 데이타를 종합하여 새로이 생성되기도 한다.

이런 여러가지 graphics 객체들의 표현공간이 되는 좌표계는 translation, scaling, rotation, reflection, 그리고 skewing등의 linear transformation을 여러 가지로 조합하여 모양과 위치를 마음대로 변경할 수 있게 되어 있다.

PS 언어는 프린트할 수 있는 글자들과 space, tab등과 같이 공간은 주기위한 글자들로 구성되어 있다. 그러므로 프로그램을 쓰거나 고치기도 쉬울 뿐만아니라 PS file을 여러 다른 시스템에서 저장하거나 전달하기도 용이하다. PS는 연산자(operator)가 연산수(operand)의 뒤에 있는 postfix notation을 사용하고, 모든 연산은 stack을 통하여 처리된다.

PS로 아주 간단한 산수 연산을 하는 과정을 살펴보자. 좌측은 PS의 syntax이고 우측은 같은 의미의 C 코드이다.

```
40 60 add 2 div          =          (40 + 60) / 2
```

Interpreter가 먼저 정수인 '40'을 만나면 그것을 operand stack에 push한다. '60'의 경우도 마찬가지로 operand stack에 push된다. 그 다음에 'add'라는 이름을 만나면 dictionary stack에서 'add'가 정의되어 있는지 검색한다. 'add'는 PS에 미리 정의된 연산자이므로 그에 해당하는 연산을 수행하게 된다. 'add'는 operand stack에서 두개의 정수를 pop하여 두 수를 더하고 그 결과를 다시 operand stack에 push한다. 나머지 부분도 이와 유사하게 실행된다. 정수 '2'를 push한 후 'div'를 수행한다. 연산자 'div'도 두개의 수를 연산수로 사용하는데 현재 100과 2가 operand stack에 있으므로 나누고 나면 50이 operand stack에 남는다.

PS의 가장 큰 장점 중의 하나는 device-independent하다는 것이다. 원하는 출력기가 어떤 종류의 어떤 해상도(dots-per-inch)이던지 상관없이 같은 PS 프로그램으로 출력할 수 있다. 또 다른 장점은 font에 있다. 기존의 bitmap(각 글자가 bit image로 구성)으로 사용하던 font에서 벗어나 각 글자를 graphics의 형태로 표현한 outline font를 채택함으로써 글자의 크기와 모양을 마음대로 변경할 수 있다. 그러나 PS는 interpreting 방식으로 프로그램을 해석하여 처리할 뿐만 아니라, 글자 인자시에도 각 글자의 윤곽선에서 bitmap을 만들어야 하므로 수행 속도가 상당히 느려서 최대의 단점으로 지적되고 있다.

2.2. Imaging Model

우리가 원하는 그림을 그리거나 글을 쓰기 위해서는 여러장의 깨끗한 종이와 연필이나 펜이 필요하다. 이런 도구가 준비되면 원하는 곳에 그림을 그리거나 글을 쓰면 된다. PS은 컴퓨터에서 이런 작업을 할 때 기존의 연필이나 펜을 대치하는 하나의 tool에 불과하다.

PS에서 한장의 종이에 해당하는 것이 current page이다. Current page는 PS의 painting 연산자에 의해서 그려진 모든 자취를 저장한다. PS는 원하는 위치에 원하는 색을 칠하는데 그전에 같은 위치에 이미 칠한 것이 있다면 그위에 덮칠을 하게 된다. 즉 유화처럼 그 전의 색이 완전히 무시된다.

실제로 원하는 위치를 지정하는 것은 current path이고 이것이 주어져야만 색을 칠할 수 있다. Current path는 요구된 모양을 점과 선과 curve등으로 나타내어 그 값을 저장하고 있다. 주의해야 할 것은 current path는 current page에 아무런 흔적도 남기지 않는다는 것이다. 어디에 어떤 모양으로 그리겠다는 것만 지정할 뿐이고 current page가 칠해지는 것은 painting 연산자들이 수행된 후이다. Painting 연산자들은 주어진 path를 일정한 두께의 선으로 그리거나, path의 내부를 원하는 색으로 칠하는 등의 일을 한다.

PS 프로그램이 출력을 받아보는 일반적인 과정은 path를 만들고, 색이나 선의 두께등과 같은 값을 지정한 후 칠을 하여 출력한다.

Current path외에 clipping path가 있는데 이는 current page내에서 실제로 칠할 구역을 제한한다. 초기치는 current page전부가 되며, 그 후에 clipping path가 변경되면 그 path 외부에 있는 것들은 버려져서 current page에 아무런 영향도 미치지 않는다. 즉 그리고자 하는 종이와 같은 크기의 종이를 원하는 부분만 잘라내고 그것을 그릴 종이위에 대고 그리면 잘려 나간 부분만 그려진다. Clipping path는 잘려나갈 부분을 정의하는 것이다.

Current page상에서 path를 정의할 때는 좌표값을 이용한다. 좌표값은 current page상에서 좌표계의 한 점을 지정하는 실수 (x, y)쌍으로 표현된다. 출력기의 종류는 다양하므로 각 출력기의 좌표계도 달라진다. PS에서는 이것을 device space라고 지칭하였다. 만약 device space를 PS 프로그램에서 직접 사용한다면 출력기가 달라지는 경우 PS 프로그램도 변경되어야 하므로 device independence를 유지할 수 없다. 그래서 PS은 user space라는 이상적인 좌표계를 정의하여 출력기에 상관없이 current page와 항상 일정한 관계를 유지하게 하였다. PS의 default user space는 원점을 좌측 하단 구석에 두고 x축은 오른쪽으로 수평하게 확장되고, y축은 위쪽으로 수직적으로 확장된다. X나 y축의 단위는 1 point 즉 1/72 inch이다. 그러므로 항상 x, y의 값은 0보다 큰 수가 된다.

PS로 한 면이 1 inch인 box를 그리는 프로그램을 살펴 보자.

```
newpath                 새로운 path의 시작
100 100 moveto          box를 그릴 시작점
172 100 lineto
172 172 lineto
100 172 lineto
closepath               box 완성
0.5 setgray             색 지정
```

```
fill                    box 내부를 칠함
showpage                출력기로 출력
```

Path를 만들고 원하는 색을 지정한 후 칠하고 마지막으로 출력기로 출력한다. 여기서 사용된 좌표값은 default user space에서의 값임을 상기하기 바란다.

3. Fonts

3.1. PostScript Fonts

출력을 위한 여러 응용 시스템에 의해 수행되는 일 중에 text를 만드는 것은 가장 빈번한 일 중에 하나이면서 또한 가장 복잡한 일 중의 하나이기도 하다. Text를 구성하는데 여러 가지 서체들이 사용되는데, 특정 크기에서의 한 서체를 font라 한다. PS에서는 기존의 출력기에서 주로 사용하던 bitmap font가 아니고, 각 글자의 윤곽선을 선과 curve로 정의하여 저장한 outline font를 채택하였다.

Font의 모든 글자들은 어떤 크기로도 확대 축소될 수 있고, 0에서 360도 어떤 각도로도 회전될 수 있다. 한 font가 전체적으로 굵어 지거나 기울어져 새로운 font를 형성하기도 한다. 모양만 변형할 수 있는 것이 아니라 내부의 색도 마음대로 변형할 수 있으며 글자의 윤곽선만을 여러가지 굵기의 선으로 그릴 수도 있다. 글자의 윤곽선은 clipping에 사용되기도 한다. 부록의 sample output에 이런 기능들이 잘 나타나 있다.

PS의 기본 font들은 주어진 글자 코드로 글자의 정의를 찾아 갈 때 상당히 융통성 있는 encoding 체계를 사용한다. 글자 코드와 글자 정의의 상관관계가 font의 일부분으로 포함되지 않고 encoding vector로 분리되어 있다. Encoding vector는 256개의 요소를 가지는 PS 배열이며, 배열은 글자 코드로 index되고 그 내용은 각 글자의 이름이다. 그래서 print할 글자가 주어지면 그 글자의 코드를 encoding vector의 index로 사용하여 글자의 이름을 얻는다. 얻은 이름을 가지고 실제 글자의 정의를 찾아 필요한 처리를 한다. 그러므로 같은 font에 대해 encoding vector만을 변경하여 전혀 다른 encoding 방법을 쓸 수도 있다. ASCII 코드를 EBCDIC 코드로 변경하는 것이 그 전형적인 예가 된다.

PS에서는 사용자가 새로운 font를 정의할 수도 있는데 위에서 언급한 encoding vector의 변경도 새로운 font 정의에 속한다. 이와 같이 일부분만 변경할 수도 있고 완전히 처음부터 새로 정의할 수도 있다. 이때 지켜야 할 몇가지 규칙만 준수하면 사용자는 자기가 원하는 font를 마음대로 정의할 수 있다.

Font를 사용하는 예를 살펴보자.

```
/Helvetica findfont          font 선택
12 scalefont setfont         크기를 정하고 그를 current font로
288 720 moveto               쓸 위치 신정 (ABC)
show                         ABC 인자
```

Helvetica라는 영문 font를 선택하여 12 point 크기로 ABC를 원하는 위치에 인자한다. Font의 크기를 정하는 단위는 현재의 user space의 단위이다.

3.2. 한글 Fonts

PS의 font는 256개의 글자로 모든 것을 표현할 수 있는 로마자 문화권에 적절하게 설계되어 있어 한글이나 한자와 같은 2-byte 문자를 적용시키기에는 불충분하다. 그래서 Adobe는 Apple에서 LaserWriterII NTX-J를 준비하는 과정에서 소위 Composite Font Extension[6]이라는 새로운 spec.을 발표했다. 이는 2-byte 코드 체계를 사용하는 일본을 포함한 한자 문화권의 여러나라를 지원하기 위한 것이다.

그러나 composite방식은 그 구체적인 구현 내용이 공개되어 있지 않고 일본어판 NTX만이 나와있는 정도여서 보급에는 아직 문제가 있다.

현재 국내에서 개발된 PS용 한글 font의 구조와 코드 체계는 각 업체마다 달라 일관성이 없다. 최근에 Adobe에서 발표한 PS level 2에 따르면 2-byte 체계에 대한 표준도 완성할 예정이므로 한글도 그에 따라 표준화되어야 할 것으로 보인다.

4. PostScript의 활용

현재 가장 많이 사용되는 방식은 최초로 PS interpreter가 소개되었던 laser 프린터 내장 방식이다. 프로그램을 수행할 interpreter가 프린터에 탑재되어 있으므로, 시스템의 종류에 상관없이 프린터와 데이타를 주고 받을 수만 있다면 프린터로 PS 프로그램을 전달하여 출력을 받아 볼 수 있다. Host 시스템에서 laser 프린터로 PS 프로그램을 보내면, 프린터는 그것을 받아 interpreter로 하여금 해석하여 결과를 출력하게 한다. 이와 같이 출력하고자 하는 프로그램을 순서적으로 보내서 처리하는 방식을 batch 방식이라 한다. 특이하게도 PS는 interactive 방식을 제공하는데 이는 사용자가 단말기를 이용해서 컴퓨터와 대화하는 것과 같다. PS 프린터를 interactive하게 지정해 놓으면 host에서는 프린터가 제공하는 prompt를 받아 원하는 PS 연산자들을 입력하여 수행할 수 있다. Host에서는 입력한 글자를 직접볼 수 있으며, 간단하게 한 글자나 한 line을 지우는 정도의 editing기능도 제공된다. 이제는 프린터가 단순히 받은 데이타를 그대로 인자만 하는 것이 아니라 하나의 컴퓨터와 같이 복잡한 연산들을 처리함을 알 수 있다.

사용자가 PS로 직접 프로그램하는 경우는 드물고 응용 S/W의 device driver가 PS 프로그램을 자동적으로 생성하는 것이 보통이다. 현재 널리 보급된 영문 S/W의 대부분이 PS의 device driver를 포함하고 있다.

PS 프로그램은 출력을 위한 연산자들로 구성된 프로그램이기도 하지만, 그 자체로도 document의 역할을 한다. PS는 printer spooler나 server, 그리고 post-processor와 같은 document manager가 사용할 있게 PS 프로그램 구조형성의 관례(structuring convention)를 정의하고 있다.[1] 가장 기본이 되는 것은 프로그램을 prologue와 script로 구분하여 prologue에는 application-dependent한 정의들을 포함시키고, script는 그 정의들을 이용하여 원하는 결과를 프로그램하는 것이다. 더구나 script가 여러장으로 구성되어 있다면, 각 장은 prologue에 대해서만 의존할 뿐이고 서로는 기능면에서 독립적이다. PS에서 이와같은 구조적 관례는 로 시작하는 comment를 이용하여 필요한 정보를 표현한다. PS interpreter는 comment를 모두 무시하므로 이런 관례에는 상관없이 주어진 작업을 수행한다.

이러한 Comment들은 PS 프로그램의 구조를 prologue, script, trailer등으로 구분하는데 사용될 뿐만 아니라 font와 같은 PS 프로그램이 요구한 자원들을 지정하는데도 사용된다. 또한 현재 프린터의 상태나 사용가능한 font, file, memory등과 같은 주어진 프린터의 특성을 알아보는 데도 이용된다. Document manager가 이런 다양한 정보를 이용하여 document들을 효율적으로 관리하게 된다.

지금까지 PostScript 언어의 구성과 일반적인 활용 방법에 대해 개략적으로 정리해 보았다. PostScript의 자세한 syntax에 대한 설명은 피하고 일반적인 특성을 중심으로 서술하였다. PostScript이해에 조금이나마 도움이 되길 바란다.

참고 문헌

[1] Adobe Systems Inc., PostScript Language Reference Manual, Addison-Wesley, 1985. [2] Adobe Systems Inc., PostScript Language Tutorial and Cookbook, Addison-Wesley, 1985. [3] Adobe Systems Inc., PostScript Language Program Design, Addison-Wesley, 1988. [4] J. Seybold and F. Dressler, Publishing From the DESKTOP, Bantam Books, 1987. [5] F. Davis, J. Barry and M. Wiesenberg, Desktop Publishing, Dow Jones-Irwin, 1986. [6] Adobe Systems Inc., PostScript Language Composite Font Extensions, Adobe Systems Inc, 1988.

부 록

HUMAN

TriGem

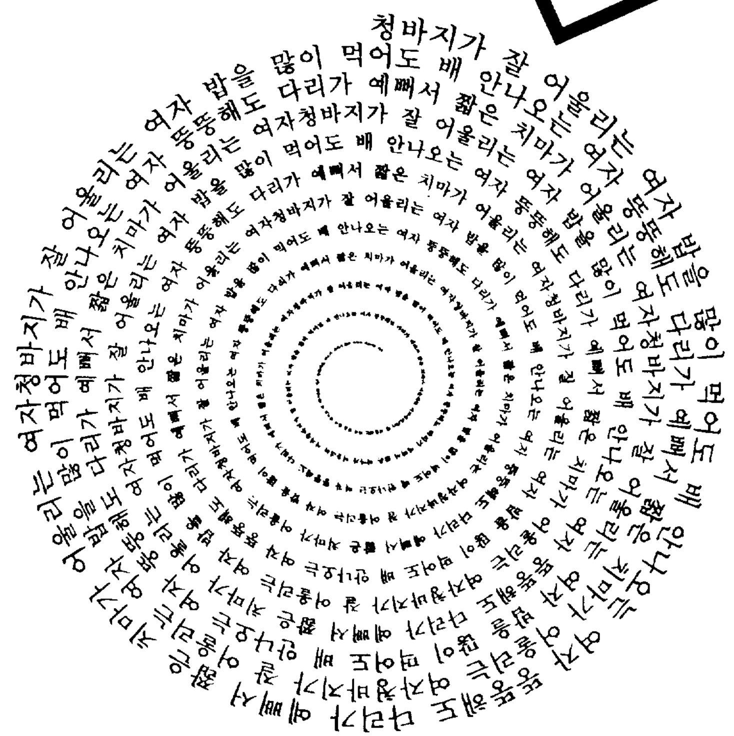
PostScript으로 작성하여 TGLBP-800BP로 출력하였습니다.
이 sample은 Adobe illustrator에서 만든것을
우리 레이저프린터로 출력한 것입니다.

ARTWORK CREATED USING ADOBE ILLUSTRATOR. ADOBE ILLUSTRATOR IS A TRADEMARK OF ADOBE SYSTEMS INCORPORATED.

Human Computers, Inc. 1990. 4

PostScript으로 작성하여 TGLBP-800BP로 출력한 것입니다.
이 sample은 한글을 다양한 크기로 회전시킨 것입니다.

아래의 소용돌이 꼴은
모든 크기의 글자를
어떤 방향으로든지
회전시킬 수 있는
PostScript의 기능을
보여줍니다.
이 모든 글자들은
한 윤곽선 서체에서
만들어진 것입니다.



Human Computers, Inc. 1990. 4