特輯 技術解說

윤곽선 글자꼴의 처리 기술 및 활용 추세

林 淳 範
(株)휴먼컴퓨터

Ⅰ. 서론 : 폰트 전쟁(The Font War)

최근 컴퓨터 산업에서 출력기의 비중이 날로 높아지고 있다. 외국의 시장 분석 자료에 따르면, 90년대 중반에는 전세계 컴퓨터 판매 시장에서 레이저 프린터의 매출이 PC 판매량을 앞지를 것이라는 전망이다. 컴퓨터 사용자는 보다 깨끗하고 보기 좋은 출력 물을 원하고 있다. 이러한 욕구를 충족시키기 위해 소음이 작고 출력의 품질이 좋은 레이저 프린터가 선호되는데, 아직은 속도와 가격 문제로 도트 프린터에 비해 대중화 되어 있지 않다. 그러나 이 점도 부품의 기술 향상과 폰트 개발로 조만간 해결이 될 것이다. 이렇듯 점점 비중이 더해지는 출력기의 가장 핵심되는 요소가 바로 폰트를 처리하는 기술이다.

1989년 봄, PostScript를 개발한 Adobe사를 둘러싸고 소위 '폰트 전쟁(font war)'이 시작되었다. 점글자꼴 레이저 프린터는 이미 70년대에 개발되었고, 윤곽선 글자꼴 방식의 레이저 프린터는 1985년 1월 Adobe사가 PostScript를 탑재한 LaserWriter를 발표함으로써 널리 보급되기 시작했다. 그런데, 이 제품에 대한 PostScript와 윤곽선 글자꼴에 대한 기술료(royalty)로 일반 기준을 훨씬 넘는 10%에 가까운 금액을 Adobe사에 지불해야 했다. 이에, 미국의 몇몇 회사들이 Adobe사의 기술을 분석하여 나름대로의 기술 개발을 추진하였다. Bitstream이라는 폰트 전문 회사는 Adobe사의 글자 그리는 기술을 파악하였다고 발표하였고, Apple사는 Royal Font라는 자체적인 글자꼴 기술을, Microsoft사는 PostScript 개발 기술을 보유하게 되었다. 이러한 상황에서 Apple사가 Adobe사의 기술 사용을 중단하고 Microsoft사와 함께 자체기술로 True Type과 Type Image를 개발하여 자사의 레이저 프린터에 사용하겠다고 발표한 것이 폰트 전쟁의 발단이다. 이에 Adobe사는 그동안 대외비로 하고 있던 PostScript의 글자 format인 Type1 format을 공개하여 여러 회사들이 자사 기술을 보다 용이하게 사용토록 하겠다고 하였다.

1989년 가을에 Adobe사는 Type1을 공개하고 Microsoft사는 True Type의 사양 초안을 발표하였다. 한편 이 사이에서 Next사는 향후 자사의 모든 워크스테이션에 Adobe사의 display PostScript를 내장하겠다고 방침을 밝혀서 Adobe사의 일시적 열세를 만회해 주었다.

1990년 봄에는 IBM이 자사 PC의 기본 OS에 Adobe사의 폰트 기술을 탑재하겠다고 발표하였다. 한편, 전세계 레이저 프린터 시장에서 60% 이상의 시장 점유율을 가지고 있는 Hewlett Packard는 윤곽선 글자꼴을 포함하는 새로운 PCL5를 발표하였다. 따라서 제2차 폰트 전쟁은 Adobe사의 Type1, Microsoft사의 True Type, 그리고 HP의 PCL5에서 사용되는 Agfa Compugraphic사의 Intellifont가 주축이 되었다.

1990년 가을 True Type의 견본이 제작되어 PostScript Type1과 비교 평가가 되면서 현재까지 우위 다툼을 벌이고 있다. 이의 결과는 True Type의 글자꼴이 실제로 탑재될 Windows 버전 3.1이 시판되는 내년 봄 이후에 결정될 것이다. 한편, 1991년 봄 Apple사는 Macintosh의 system7부터 Adobe의 Type1 기술을 탑재하겠다고 발표하여 폰트 전쟁은 더욱 혼미에 빠져 있는 상황이다.

지금까지, 세계의 유수 컴퓨터 업체들이 프린터 시장에서 글자꼴 기술을 중심으로 치열한 경쟁을 하고 있음을 보았다. 다음으로, 글자꼴 기술이 어떤 것인지, 그리고 이것이 어떻게 활용이 되는 가를 살펴보도록 한다. 2장에서는 글자꼴의 세 가지 형식에 대해 분류하고 각각

의 장단점과 활용 가능성에 대해 설명하였다. 3장에서는 특히 윤곽선 글자꼴에 대해 설계와 출력 생성시 사용되는 기술에 대해 정리하였고 글자의 품질을 높이는 기법에 대해 언급하였다. 마지막 장에서는 글자꼴 기술의 향후 방향과, 한글을 처리하는 데 있어서의 글자꼴 기술의 연구 필요성에 대해 설명하였다.

Ⅱ. 글자꼴의 형식과 응용분야(Font Classification and Applications)

1. 글자꼴 형식에 대한 고려 사항

컴퓨터로 표현되는 글자꼴 형식에는 기본적인 표현 단위의 대상에 따라 크게 세 가지로 분류할 수 있다.[5] 글자를 그림 또는 화상(image)으로 간주하는 점글자꼴(bitmap font), 윤곽선 글자꼴(outline font), 글자골격의 위상(topology)으로 표현하는 구조적 글자꼴(structural font)의 세 가지이다. 글자꼴 형식을 선택할 때의 주요 관점은 기존 글자꼴의 품질을 유지하면서 어떻게 컴퓨터 기술의 장점을 살릴 것인가에 있다. 이때, 제작시와 글자 출력 생성시의 두 가지 측면에서 다음과 같은 사항들을 고려해야 한다.

제작 측면에서 보면,

- 설계자가 얼마나 쉽고 효율적으로 설계하여 소요시간이 얼마나 걸리는가 고려해야 한다. 이 때에는 글자를 제작하는 시스템의 효율성도 영향을 크게 미친다.
- 글자의 원도나 종이에 그린 초안을 카메라나 스캐너로 입력받아 얼마만큼 자동으로 설계가 가능한지도 고려해야 한다. 이는 제작할 글자의 수가 매우 많을 때 중요한 사항이 된다.
- 글자 데이타의 기억 장소 사용량도 중요하다. 메모리 가격의 하락으로 영문자에 대해서는 기억량의 중요성이 줄어들고 있지만 글자수가 많은 한글이나 한자의 경우에는 여전히 중요한 고려 사항으로 남아있다.

출력 생성의 측면에서 보면,

- 출력 결과의 품질이 매우 중요하다. 품질은 출력기기의 해상도나 원래 글자의 모양과 밀접한 관계가 있지만, 글자의 품질은 출력기의 글자 처리기능에 따라 가장 큰 영향을 받는다.
- 출력기기의 성능에 영향을 주는 또 하나의 큰 요소가 글자 출력시 생성 속도이다. 제품에 사용할 때 생성 속도가 느린 글자꼴 형식에는 빠른 기계를 써야만 제대로 실용화를 할 수가 있다.
- 글자꼴의 여러가지 변형에 대한 가변성도 고려해야 한다. 크기의 변화, 회전, 기울임의 변화 등 여러가지 변형에 대해 글자의 품질이나 생성 속도가 표현 형식에 따라 다르게 된다.

2. 점글자꼴(bitmap font)

대부분의 프린터는 글자를 점으로 찍어서 출력을 하는데, 이 점들을 그대로 표현한 방식이 점글자꼴이다. 이는 가장 간단한 방식으로, 글자의 내부에 해당하는 부분을 1로 하고 외부를 0으로 하여 한글자에 해당하는 사각형을 점행렬식(dot matrix)으로 기억한다.

이 방식은 설계할 때의 모양과 출력된 글자 모양이 같으므로 설계자의 의도를 그대로 표현할 수 있다. 그러나, 작은 점 하나가 전체 글자의 균형에 미치는 영향 등을 고려해서 설계해야 하기 때문에 제작 시간이 길어질 뿐만 아니라, 글자꼴 정보를 저장하기 위한 기억량도 매우 커야 한다. 기억 용량을 줄이기 위한 방법으로는 연속점 길이 방식(run-length coding)이나 방향점 방식(chain-link coding)등과 같은 좀 더 효율적인 표현 방식을 쓰고 있다.[3]

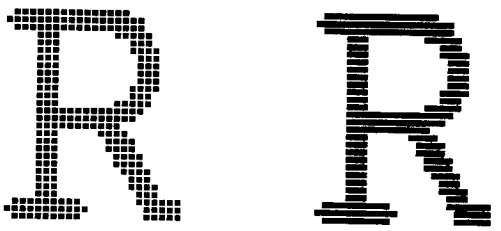


그림 1. 일반 점글자(a)와 연속점 길이 방식의 점글자(b)

해상도가 낮은 대부분의 점행렬식 프린터에서는 이 방식의 글자꼴을 채택하고 있는데, 이는 해상도의 한계로 하나의 글자꼴을 표현하는 점의 수가 적으므로 설계 시간이나 기억량이 크게 문제되지 않기 때문이다. 크기나 폭의 변화는 정수배의 확대나 축소가 가능한데, 글자를 확대할 경우 윤곽이 거칠어진다. 또한 기울임이나 굵기의 변화 역시 처리할 수 없으므로, 결국 모든 크기의 글자를 가지고 있어야 한다.

점글자꼴은 제작과 처리 과정에서 매우 간단하기 때문에 지금까지 많은 제품에 사용되어 왔다. 컴퓨터 모니

터, 도트 프린터, 레이저 프린터, 그리고 최근에는 잉크젯 프린터 등 raster device에서 주종으로 사용되어 왔다. 그러나 최근 고가의 고품질 레이저 프린터 등에서는 윤곽선 서체 사용이 늘어나고 있다.

3. 윤곽선 글자꼴(outline font)

윤곽선 글자꼴은 글자의 윤곽을 여러 부분으로 나누어 직선, 원호, 자유 곡선 등으로 표현한다. 자유 곡선으로는 3차 운형(cubic spline)곡선, 베지어(Bezier)곡선, B운형(B-spline)곡선 등이 주로 사용되며, 최근에는 원추 운형(conic spline)곡선도 사용된다.[5]

실제 폰트 전쟁의 주역인 Adobe사의 PostScript Type1은 3차 베지어 곡선으로 표현하고 Microsoft사의 True Type은 2차 B운형 곡선을 사용한다. 세계적인 서체제작시스템 개발 회사중의 하나인 URW에서 개발한 IKARUS시스템과 Agfa Compugraphic의 Intellifont는 3차 운형 곡선을 주로 사용한다.

윤곽선 글자꼴을 설계하는 것은, 그 특성상 제도기를 이용하여 손으로 글자를 설계하는 기존의 방법과 유사한 과정을 거친다. 그러나 이를 위한 윤곽선 편집기가 필요하고, 그 시스템의 기능에 따라 작업 효율이 매우 달라진다. 한편 종이에 그려진 원도가 있는 경우에는 이를 크기가 큰 화상으로 입력시킨 후 윤곽점을 따라가며 자동으로 곡선과 직선으로 계산해 주는 프로그램이 있어서 수작업의 양을 줄일 수 있다.

윤곽선 글자꼴은 글자의 크기나 기울기, 회전 등에 대해 자유롭기 때문에 하나의 크기에 대한 데이타만 필요하므로 점글자꼴에 비해 기억용량이 매우 감소한다. 글자꼴의 변화는 사진 식자에서 렌즈가 처리해주는 부분까지 가능하다. 그러나, 획의 굵기 변화등 글자꼴 자체에 대한 변화는 처리할 수 없다.

출력 글자를 생성할 때는 일단 출력하는 크기에 맞춰 윤곽선의 직선이나 곡선을 점으로 그려주고 그 내부를 채우는 과정이 필요하므로 속도가 느려진다. 이러한 계산 과정 때문에 70년대까지는 실용화되지 못하다가 80년대 들어와 프로세서의 발달로 레이저 프린터 등에서 쓰이고 있다.

한편, 글자의 품질은 크기가 큰 글자이거나 고해상도의 출력기일 수록 정교해지지만 작은 글자에서는 계산상의 오차로 인해 점글자보다 품질이 떨어진다. 그래서 윤곽선 데이타에서 계산 도중 보정을 해주어야 하는데, 이를 소위 힌팅(hinting)이라고 한다. 이 개념을 최초로 도입한 제품은 Adobe사의 PostScript로, 글자를 그릴 때 내부적으로 계산하여 처리된다. 폰트 전쟁에서 가장 핵심되는 기술이 글자의 생성 속도와 윤곽선 글자의 품질을 향상시키는 바로 이 힌팅기술이다.

현재 윤곽선 글자꼴은 90년대 들어와 여러 레이저 프린터에 널리 보급되고 있고 인쇄용 출력기에도 많이 쓰이고 있다. 영문자 권에서는 프린터 뿐만 아니라 화면에서도 윤곽선 글자를 출력해주는 프로그램이 개발되어 90년도부터 실제 상품으로 제작되고 있다. 여기에는 Adobe사의 ATM, Bitstream사의 FaceLift, Publisher사의 Powerpack등이 있다.

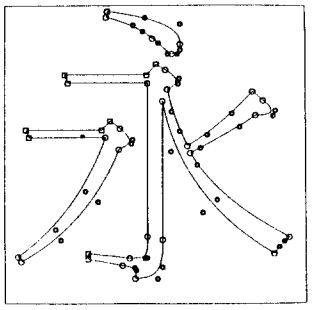
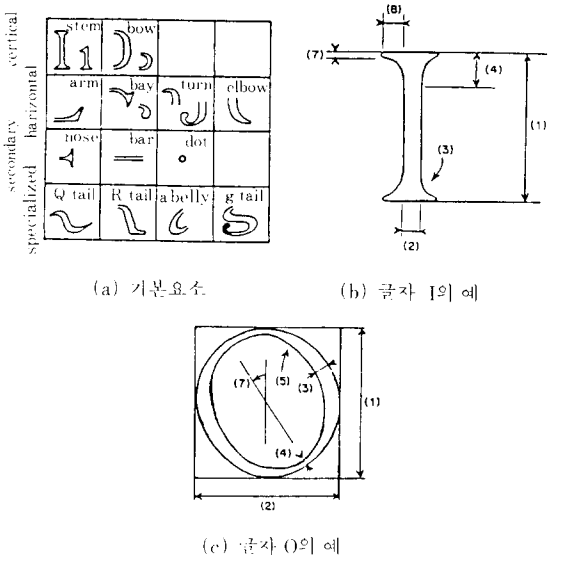


그림 2. 3차 베지어 곡선을 사용한 윤곽선 글자

4. 구조적 글자꼴(structural font)

구조적 글자꼴은 윤곽선을 이용한 구조적 글자꼴과 중심선(skeleton)을 기초로 하는 글자꼴의 두 가지로 구분할 수 있다. 이들은 개별적인 글자보다 한 벌 전체의 글자꼴 특성을 고려하여 글자를 설계할 수 있는 표현 방식이다. 따라서 기억량이 매우 줄어들고 글자의 설계나 글자체 변화가 매우 용이해지는 반면, 생성 속도가 매우 느려진다. 외국에서 개발된 대표적인 시스템으로는 전자의 경우 Ph. Coueignoux의 CSD시스템이 있고,[3] 후자의 경우 Stanford대학의 D.E.Knuth가 개발한 Metafont 시스템이 있다.[4]

Ph. Coueignoux는 CSD에서 영문자에 대해 그 구조를 분석하여 그림3과 같이 50여개의 기본 요소를 추출하고, 각 글자는 이로부터 조합하여 설계하도록 하였다. 기본 요소는 각기 특징매개변수 목록(shape parameter list)을 가지고 있으며, 그 윤곽선은 원추 운형 곡선으로 표현된다. 글자를 조합할 때는 기본 요소의 위치값과 특징 매개 변수에 대한 값을 주고, 각 기본 요소들의 윤곽선은 원추 운형 곡선 사이의 보간에 의해 서로 연결되도록 하였다. 또한 글자꼴 전체에 대한 글자꼴 변화 변수(font variation parameters)를 가지고 있어서 여러가지 글자체로 변형이 가능하다.



(a) 기본요소 (b) 글자 I의 예

(c) 글자 O의 예

그림 3. CSD의 예[2]

글자꼴 모양을 결정하는 요인으로는 크게 두 가지가 있는데, 펜의 모양과 펜의 움직이는 자취가 바로 그것이다. 이러한 펜의 설계와 중심선의 자취 설계를 독립적으로 분리 시키면 글자 모양의 설계를 보다 다양하게 할 수 있는데, 이러한 개념을 최초로 도입한 것이 Metafont시스템이다. Metafont에서는 베지어 곡선과 3차 운형 곡선으로 표현된 글자의 중심선을 따라 펜이 움직일 궤적을 이미 설계된 펜의 모양에 따라 점(pixel)으로 채워서 글자를 생성한다. 이 방법은 사람이 글씨를 쓰는 과정과 비슷하므로 글자의 설계가 쉽고, 기억 용량이 더욱 작아지며, 펜의 교체에 의해 여러 가지 글자꼴의 효과를 낼 수가 있다.

구조적 글자꼴은 설계할 때나 데이타를 저장할 때 매우 효율적이고 출력 결과의 품질도 향상되지만, 처리과정이 복잡하기 때문에 생성 속도가 큰 문제가 된다. 현재 출력기에서 곧바로 사용되는 경우는 없고, Metafont 또는 글자 설계까지만 활용되고 있다.

그림 4. Metafont의 글자 예[4]

Ⅲ. 윤곽선 글자꼴의 처리기술(Outline Font Processing Technology)

1. 글자 제작시 용이도(design efficiency)

윤곽선 글자를 제작하는 경우는 크게 두가지로 나눌 수 있다. 디자이너가 윤곽선 글자 설계 시스템을 사용하여 새로운 글자꼴을 직접 설계하는 것과 기존의 종이에 그려진 원도를 윤곽선 글자꼴로 전산화하는 것이다.

현재 전자와 같은 경우를 위한 소프트웨어는 다수가 있는데 이들은 글자를 설계할 수 있는 일종의 그래픽에디터라고 볼 수 있다. 글자 설계 작업의 효율성은 소프트웨어의 기능 여부에 좌우되는데, 이때 중요한 판단 요소는 다음과 같다.

- 윤곽선의 편집 기능이 편리한가 : 구간점의 이동, 추가, 삭제, 특성 변경 등의 기능이 다양한 지의 여부
- 글자의 보여주기 기능이 편리한가 : 글자의 크게보기, 작게보기, 이동, 기준선, 생성결과, 분장에서의 사용결과등의 기능이 다양한 지의 여부
- 반복부분의 작업량 축소가 편리한가 : 글자의 일부 또는 전부의 복제 및 저장기능이 가능한 지의 여부
- 글자의 선택이 편리한가 : 원하는 글자를 자유자재로 선택하여 설계하는 기능이 다양한 지의 여부

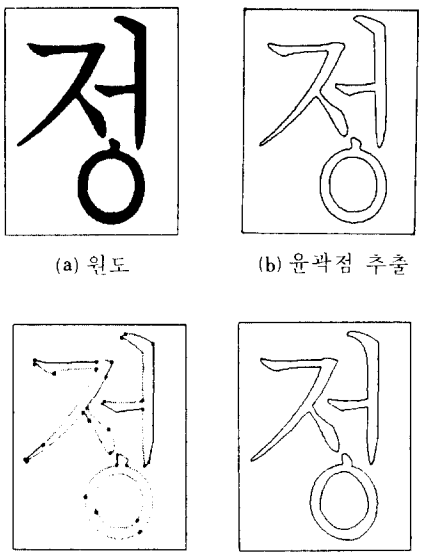
원도를 전산화하는 데에 필요한 시스템은 위에서 열거한 기능 외에 원도를 입력하는 기능이 필요하다. 한글이나 한자와 같이 글자수가 많은 경우에는, 입력된 원도로부터 자동으로 윤곽선을 생성해주는 기능이 있으면 작업 시간이 대폭 단축된다. 이를 보통 윤곽선의 자동추출(auto-tracing)이라고 하는데, 계산 과정은 다음과 같다.[6]

(1) 윤곽점을 추출한다 : 입력된 원도의 화상으로부터 윤곽점에 해당되는 chain code를 추출하는 과정으로, noise filtering기술을 적용하면 화상 입력 과정에서 생긴 error를 줄일 수 있다.

(2) 윤곽선의 구간을 분할한다 : 윤곽점 chain code에서 직선에 해당하는 구간과 곡선에 해당하는 구간으로 분할하고, 이들 구간점 중에서 글자의 특징점을 결정한다. 특징점으로는 corner point, 기울기 연속점, 극단점, 삐침점등이 있는데 이들을 찾아낼 때 사용되는 threshold value는 알고리듬과 서체에 따라 다르다.

(3) 곡선의 근사치를 구한다 : 글자의 윤곽선중 곡선 구간에 대해 해당곡선의 control point를 계산하는데, chain code와 구간점의 특징 정보를 이용한다. 연속점의 경우는 기울기를 미리 제한하고 least square fitting 방법

을 적용하여 곡선 근사를 한다. 실제 윤곽점과 계산상 곡선 사이의 오차합이 적정 범위내에 들어올 때까지 이 과정을 반복한다.



(a) 원도 (b) 윤곽점 추출

(c) 구간분할 (d) 곡선근사

그림 5. 자동 추출 과정

2. 출력 생성 속도(rendering speed)

윤곽선 글자가 레이저 프린터와 같은 상품에서 사용될 때 가장 중요한 요소 중의 하나가 출력속도이다. 속도는 주로 프로세서에 의해 좌우되는데, 같은 프로세서라도 알고리듬에 따라 속도 차이가 심하다. 이러한 이유로 최근 글자를 그리기 위한 곡선 생성과 내부채우기 알고리듬에 대한 연구가 업계를 중심으로 활발히 진행되고 있다.

1) 곡선의 생성(curve rendering)

PostScript와 함께 널리 쓰이는 곡선은 3차 베지어 곡선으로, 다음 두 가지 방법이 곡선을 점으로 변환시켜 주는 가장 대표적인 곡선 생성 알고리듬이다.[6]

(1) Parameter 계산방법

일반적인 3차 베지어 곡선에 대한 계산 공식은 다음과 같다.

하나의 곡선 구간에 대해 양 끝점 P_1, P_4, 중간 control point P_2, P_3가 주어졌을 때,

$$P(t)=(1-t)^3P_1+3t(1-t)^2P_2+3t^2(1-t)P_3+t^3P_4,\quad 0\le t\le 1$$

t의 값이 0에서 1까지 변화하므로 n등분을 하여 각각의 t값에서의 P(t)값을 구하고, 이들을 연결하면 n개의 연속된 직선이 된다. 각 직선에서 정수 좌표에 해당하는 연속점들을 구해 나가면 원하는 곡선에 해당하는 점이 된다. 이때, 구간 등분 n을 작게하면 계산 속도는 빠르지만 곡선의 질이 떨어지고, 이와 반대로 n을 크게하면 곡선의 질은 좋으나 중복되는 직선이 많으므로 계산 속도가 느려진다.

(2) Midpoint subdivision 방법

주어진 네 개의 베지어 곡선점들의 중간값들을 구하면 원래의 곡선을 두개로 나누어서 정의할 수 있는 원리를 이용하는 것이다. P_1, P_2, P_3, P_4가 주어졌다면,

$P_{12}=(P_1+P_2)/2$, $P_{23}=(P_2+P_3)/2$, $P_{34}=(P_3+P_4)/2$,
그리고, $P_{123}=(P_{12}+P_{23})/2$, $P_{234}=(P_{23}+P_{34})/2$
$M=(P_{123}+P_{234})/2$

이때, P_1, P_{12}, P_{123}, M은 앞쪽 절반 곡선의 베지어 곡선점이 되고, M, P_{234}, P_{34}, P_4는 뒷쪽 절반 곡선의 곡선점이 된다. 이와같이 중간값들의 계산과 곡선 분할 과정을 계속하여 곡선 자체가 하나의 점에 수렴할 때까지 반복하면 원하는 곡선이 구해진다. 이 방식은 덧셈 연산과 shift (2로 나누기) 연산만 사용하므로 계산속도가 매우 빠르다.

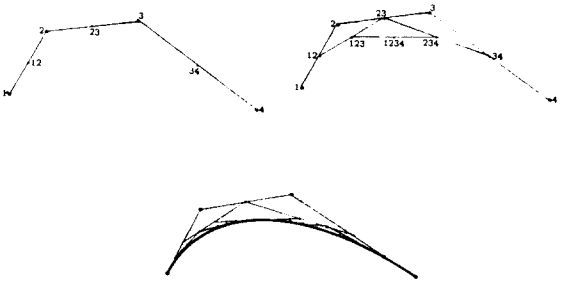


그림 6. Midpoint subdivision 방법의 베지어 곡선

2) 내부 채우기(filling)

곡선에 대한 계산을 하고 나면 그 내부를 채워야 하는데, 여기에는 크게 두가지 방식이 있다. 하나는 곡선 계산과 내부채우기를 동시에 하는 방식으로 보통 scan

conversion이라 하고, 다른 하나는 윤곽선에 해당하는 모든 점들을 buffer에 그린 후 그 내부를 채우는 filling 방식이다.

Scan conversion은 곡선을 계산하여 직선으로 변환시킨후, 윤곽선에 해당하는 다각형에 대해서 한다. 우선 다각형의 모든 변에 대해서 y방향의 시작점을 기준으로 sorting을 한다. y값을 하나씩 증가시키면서 각 scanline에 걸치는 변들만 골라내어 이들 사이를 점으로 채워주면 전체 글자가 완성된다. 이때 윤곽선의 회전방향을 고려하면 non-zero winding 방식이라 하고, 고려하지 않으면 even-odd방식이라 하는데, 그 결과는 그림 7과 같이 다르게 나타난다.[11]

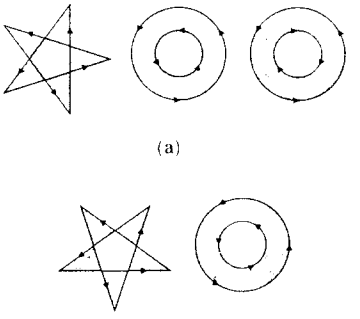


그림 7. Non-zero winding 방식(a)과 even-odd 방식(b)

일반 filling방식은 일단 buffer에 윤곽점들을 그린 후 그 내부를 찾는다. 내부영역에 해당하는 하나의 점이 주어지고 여기부터 채워나가는 seed filling 방식은 글자를 생성할 때는 그러한 점이 주어지지 않으므로 적용할 수 없다. 따라서 buffer의 내용을 scanline별로 추적하여 이미 그려진 윤곽선의 점과 점사이를 even-odd방식으로 채워준다. 이때 윤곽선의 상하 극점에 해당하는 부분에서는 윤곽점이 하나가 되므로 이를 반드시 검사해야 하는데, 이를 parity check라 한다. 이 과정에 대한 계산을 효율적으로 하기 위해 CLAG, edge flag, fence fill등 여러가지 알고리듬이 개발되어 있다.[11]

LBP의 경우 점글자발생용으로는 주로 Motorola의 M68000계열의 프로세서가 사용되는데, 위와 같은 윤곽선글자 처리에는 성능이 부족하다. 요즘에는 이를 위해 graphics성능이 강화된 National Semiconductor의 CG16, GX32 또는 image processing용 DSP와 같은 보조 프로세서를 쓰기도 한다. RISC 프로세서가 LBP용으로 이용되는 추세도 있는데, 여기에는 MIPS의 R3000계열, AMD의 29000계열, Intel사의 i960계열, Motorola의 88100등이 있고, printer 기능이 추가된 Weitek사의 XL8220이라는 프로세서도 있다. 한편으로는 윤곽선 데이타로부터 글자 image를 최종 생성하는 과정까지를 ASIC으로 제작하여 속도를 향상시키기도 한다.

3. 출력글자의 품질(output quality)

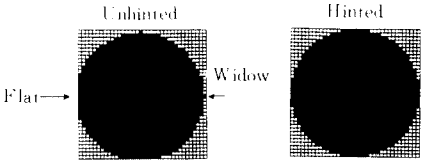
윤곽선 글자의 품질을 결정하는 중요한 요소는 곡선과 내부채우기 알고리듬의 성질이다. 그러나 윤곽선 글자는 크기가 클수록 효과가 증가되고, 오히려 작은 글자에서 출력 품질이 문제가 된다. 이는 윤곽선의 좌표 계산 도중의 truncation error와, 점으로 변환시키는 digital화 과정의 분해점이 작은 글자일 수록 큰 영향을 미치기 때문이다. 이로 인해 생기는 현상에는 다음과 같은 것들이 있다.

- Unsmooth curve : 곡선이 정수 좌표 사이에서 지나가는 위치에 따라 뾰족한 섬(widow pixel)이나 편평한 점(flat run)이 나올 수 있다.
- Uneven stem : 획의 굵기가 일정치 않거나 작은 글자인 경우 수평, 수직획이 사라지는 경우가 있다.
- Drop-out : 경사진 획이나 곡선 부분의 중간이 끊어지는 경우가 생긴다.
- Inconsistent serif : 획의 serif 모양이 글자나 위치에 따라 일정치 않게 출력 결과가 나온다.

이와같은 digitization problem을 해결하는 방법을 힌팅이라고 하는데, 이는 소위 폰트전쟁의 가장 핵심적인 기술이다. 힌팅기술은 현재 서체 기술을 보유하고 있는 회사마다 기법의 차이가 심하며, 대부분 글자를 생성하는 도중에서 계산으로 처리된다. 영문자 폰트에 사용되는 힌팅의 대표적인 기법은 다음과 같다.

- Height/alignment compensation : 영문자에서는 기준선이 정의되어 있는데, E와 같이 편평한 글자는 그 높이가 같고 A와 같이 뾰족한 글자는 약간 높게 만든다. 그러나 어느 기준 이하의 작은 글자에서는 섬 하나의 차이가 크므로 이를 무시하고 높이를 모두 같게 해준다.
- Stem control : 수평, 수직획이 사라지거나 굵기에 차이가 나는 것을 방지하기 위하여 굵기 조절과 위치 조절을 한다.
- White space control : 글자에서는 획뿐만 아니라 획사이의 공간도 중요하다. 획들이 서로 붙지 않게 하고 간격이 일정하도록 하기 위해 획의 위치를 조절한다.

- Drop-out control : 하나의 획을 이루는 점들끼리 연결이 반드시 되도록, 획의 진행방향을 따라 검과 검사이에 공간이 생기면 이를 메꾸어준다.
- Curve control : 획의 serif 모양이나 곡선의 모양을 부드럽고 일정하게 하기 위하여 정수 좌표사이에서 곡선이 지나가는 위치를 일정하게 한다.

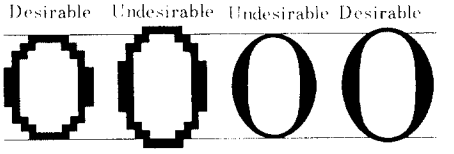


(a) Window pixel, flat run

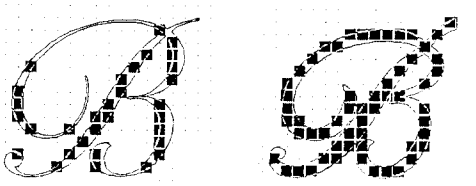
(b) 인쇄 출력 결과

(c) Uneven stem width

(d) 힌팅 결과



(e) Height compensation



(f) Drop-out control

그림 8. 힌팅의 예

Ⅳ. 글자꼴의 향후 기술(Future Technology)

1. 표현 방식의 발전(advanced representation method)

컴퓨터 기술의 계속적인 발전은 윤곽선 글자꼴의 속도 문제를 점진적으로 해결해 줄 것이다. 따라서, 향후 윤곽선 글자꼴에 대한 연구는 글자꼴의 품질을 높이기 위한 기술과 기억용량을 줄이기 위한 표현 방식에 더욱 집중될 것이다. 그동안 계산의 복잡도 때문에 실용화되지 못했던 구조적 글자꼴도 몇 년내로 실용화될 것이다.

현재 윤곽선 글자꼴의 표현 방식에 대해서 여러 가지 새로운 시도가 연구되고 있는데, 그 중 하나의 예가 1991년 봄 Adobe사가 발표한 multiple master typeface라는 기술로서, 앞으로 PostScript에 활용한 것이라고 한다. 글자의 크기 뿐만 아니라 획굵기, 폭, 글자꼴 등에 대해서도 자유자재로 변형을 시킬 수가 있는데, 현재 견본까지만 제작 실험을 한 상태이다. 획굵기나 글자의 모양 등이 다른 두 글자에 대해 모양에 대한 보간을 하여 원하는 중간 글자를 얻을 수 있다. 이렇게 되면 기본적인 몇 가지 글자꼴의 데이타만 출력기에 저장시켜도 여러가지의 중간 글자꼴을 출력할 수 있게 된다.

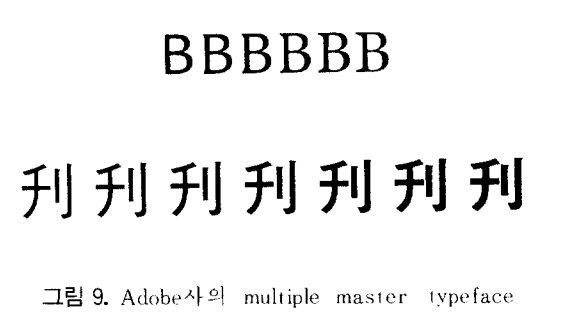


그림 9. Adobe사의 multiple master typeface

한편, 획을 중심으로 윤곽선을 표현하는 방법도 많이 연구되고 있다. 글자에서 획들을 분리하여 각 획의 윤곽선을 별도로 저장하기도 하고, CSD에서 같이 획의 일부만을 획요소로 따로 저장하기도 한다. 한 벌의 글자꼴에서는 같은 획이나 획요소가 여러번 반복 사용되므로, 이러한 방식을 이용하면 기억용량을 줄이고 제작 시간을 단축할 수 있으며 한 벌의 글자꼴 내에서 일관성을 유지하기가 쉬워진다.

2. 한글의 특수성(Hangul characteristics)

글자꼴 기술은 그것이 적용되는 글자꼴의 모양에 기

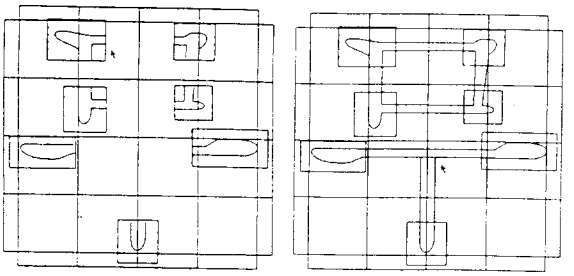


그림 10. 휴먼컴퓨터의 TypeMaster

초를 두고 있다. 현재 개발되어 있는 글자꼴 기술은 대부분이 영문자권의 것인데, 영문자와 동양문자는 근본적으로 특성이 다르다. 영문자는 대부분 켄 글씨나 제도 글씨에 기초를 두고 있고, 한글이나 한자의 글자꼴은 붓글씨에 기초를 두고 있다.

특히 힌팅 같은 기술은 컴퓨터가 글자의 구조를 파악하여 처리하는 것이므로 한글에 적용했을 때 그다지 효과가 없는 경우도 있다. 따라서 한글을 처리하는 글자꼴 기술은 현재의 영문자 글자꼴 기술 외에 추가로 처리를 해주거나 한글 특성에 맞는 새로운 기술을 개발하여야 한다. 이를 위해 다음과 같은 문제점에 대한 연구가 있어야 한다.

- 모아쓰기의 처리 : 이는 한글 고유의 특성이므로 우리가 개발하여야 한다. 모아쓰기 글자의 출력 결과에 대한 품질을 높이기 위해 다양한 모아쓰기 프로그램이 실험, 개발되어야 하고 이를 제작하는 시스템도 개발되어야 한다.
- 한글 특성에 맞는 출력 생성 알고리듬 : 한글의 여러 가지 서체에 맞도록 힌팅 등 글자 생성 알고리듬에 대해 나름대로의 기술을 개발하여야 한다.

표현 및 저장 방법 : 한글이나 한자는 영문자에 비해 글자의 수가 20~50배 정도로 많기 때문에 기억용량도 매우 크고, 경우에 따라 출력 속도가 줄어들 수도 있다. 따라서 한글의 구조에 적합한 표현 방식이 이 문제의 해결에 기여를 할 것이다.

參 考 文 獻

[1] Adobe Inc., 휴먼컴퓨터 역, PostScript 참조 매뉴얼, 오롬 정보처리, 서울, 1991.

[2] P.Coueignoux, "Character generation by computer", *Computer Graphics and Image Processing*, vol.16, no.3, pp.240-269, July 1969.

[3] J.D.Foley, A.Van Dam, S.K.Feiner, J.F.Hughes, *Fundamentals of Computer Graphics*, 2nd Ed., Addison-Wesley, Reading, Mass., 1990.

[4] D.E.Knuth, *Computers & Typesetting/Volume C : The METAFONT Book*, Addison-Wesley, Reading, Mass., 1986.

[5] 임순범, "글자체 설계 및 자동 생성 시스템의 개발", 폰트 개발과 표준화 워크샵 발표논문집, 한국정보과학회 SW공학연구회, pp.3-6, 1989. 4.

[6] 임순범, 박찬종, "한글 윤곽선 서체의 대량 제작 시스템", 우리말정보화잔치 '91 논문집, 국어정보학회, pp.379-386, 1991. 6.

筆 者 紹 介

林 淳 範

1959年 1月 30日生

1982年 2月 서울대학교 계산통계학과(학사)

1983年 8月 한국과학기술원 전산학과(석사)

1983年 9月~현재 한국과학기술원 전산학과(박사과정)

1989年 2月~현재 (주)휴먼컴퓨터 이사

주관심분야 : 컴퓨터 그래픽스, Font, LBP, 전자출판