인터넷에서의 개선된 벡터라이징 기법에 관한 연구

김용호* · 이윤배*

A study of improve vectorising technique on the internet

Young-Ho Kim* · Yun-Bae Lee*

요 약

현재 대부분의 웹디자이너들은 비트맵 그래픽을 사용하여 고정된 포인트 사이즈로 하이 퀄리티를 보장하고 있지만 이는 파일 크기와 유연성에 결점을 가지고 있다. 특히 배너문자나 광고문자에 하이 퀄리티를 제공하기 위해서는 반드시 다른 비트맵 에디팅 프로그램을 사용해서 작업한 후, 비트맵 데이터로 HTML 문서에 첨가하는 방식을 따를 수 밖에 없다. 또한 HTML 문서 자체적으로 글꼴을 단순하게 출력하는 방법 이외에, 단순한 blink, underline, bold, italic 을 제외한 글꼴을 직접 제어하는 HTML Tag 또한 가지고 있지 않기 때문이라고도 할 수 있다. 때문에 폰트의 아웃라인 데이터를 이용한 효과나 외곽선 패턴 분할 같은 작업을 위해서는 벡터 에디팅 프로그램과 이미지 에디팅 프로그램, 그리고 최종적으로 HTML 문서에 삽입하는 번거로운 과정을 거쳐야만 하는 문제에 직면하게 된다. 따라서 본 논문에서는 HTML 문서의 폰트에 몇 가지 태그를 새롭게 제시함으로써, 폰트에 더욱 다양한 효과를 줄 수 있는 방법을 제안한다. 제안된 방법은 텍스트 정보 저장시 단순한 제어점과 외곽선 정보만을 가지고 화면 출력하기 때문에 웹브라우저 상에서 인쇄물과 동일한 품질의 한글 문자 표현이 가능하며, 이종의 플랫폼에 상관없이 정확한 문자 표현, 다양한 효과로 문자 표현이 가능하다.

ABSTRACT

Currently, most web designers guarante high quality using bitmap graphics as fixed font size, but that has defects about file size and flexibility. Especially, to provide high quality of banner and advertise characters, after you should use a bitmap edit program, and then we should follow the method we add that program to HTML documents as bitmap data. In this study, as I show a couple of new tags in front of HTML documents, I show methods which can be presented diverse effects. When text information are stored, because we print out a screen with simple control points and outside information, it can be possible for us to express the same quality of Hangul characters like printed documents in a web browser. Regardless of the second class of platform, we can make it possible the character expression with exact character expressions and diverse effects.

* 조선대학교 전자계산학과
** 광주광역시 및 전라남도 지역 정보화 추진위원회
접수일자 : 2002. 4. 10

Ⅰ. 서 론

지난 몇 년 동안 인터넷 사용의 폭발과 웹 사이트의 증가는 출판 방식을 크게 변화시키고 있다. 단순한 HTML 도큐멘트에서부터 보다 인터랙티브(Interactive)하게 디자인된 예술적인 로케이션까지 다양하게 발전하고 있다. 때문에, 인터넷상에서 올바른 정보 전달, 효과적인 정보 전달, 신속한 정보 전달이 갈수록 요구되고 있는 상황이다. 그러나 국내의 경우 2-Byte 문자를 사용하고 있는 관계로 사용하는 컴퓨터의 운영체제나 플랫폼의 환경에 따라 올바른 문자 표현에 제약을 받고 있으며, 인쇄물과 동일한 품질의 문자 표현에 어려움을 겪고 있다[12]. 현재의 웹 페이지 저작 도구(Authoring tool)의 주요 결점 중의 하나는 디자이너들이 원하는 서체를 선택할 수 없다는 것과 방문객들이 언제나 동일한 폰트로 볼 수 없다는 사실이다. 이것은 폰트 공급업체, 오쏘링 툴 제조사, 고성능 오퍼레이션 시스템과 브라우저 공급자들이 현재 당면하고 있는 문제이기도 하다. 때문에 폰트 임베딩(embbeding : 실제 폰트를 데이터에 삽입하는 것)은 웹 페이지에서 폰트의 충실성과 고품질을 보장하는 최상의 해결책으로 부상하고 있는데, 트루타입(true type)과 타입 1 폰트 모두 그 목표가 되고 있다. 적절히 힌팅(hinting)된 트루타입은 스크린 상에 높은 퀄리티(Quality)를 제공하며 인쇄와 출판에서의 산업 표준인 타입 1도 활발하게 웹으로의 발전을 도모하고 있다. 지금까지, HTML 웹 페이지에서 폰트를 규정하는 것은 어려운 문제였다. 디자이너는 폰트를 선택하여 HTML 페이지에 지정할 수 있었으나 만약 방문객이 자신의 시스템에 사용할 수 없는 폰트는 자동으로 다른 서체로 디폴트(default)되게 된다. 브라우저의 폰트 디폴트 기능은 또한 사용자가 변경할 수 있으므로 웹 페이지 디자인에서 계획했던 것과 다른 폰트를 선택할 수도 있다. 웹 디자이너가 자신이 원하는 폰트를 방문객이 볼 수 있도록 하는 유일한 방법은 비트맵 포맷으로 폰트를 만드는 것이었다. 그러나 이것은 데이터 파일이 너무 크고 유연하지 못하며 텍스트 전용 브라우저일 경우는 페이지에 나타나지 않는다는 문제가 발생한다. 반면에, 폰트 임베딩은 실제 폰트가 웹 도큐멘트에 지정되므로 이러한 문제점들은 해결할 수 있으며 방문객의 시스템에 인스톨되어 있지 않아도 스크린에서 볼 수 있게 된다[5]. 폰트 임베딩으로 방문객이 얻는 또 다른 이점은 고품질이다. 도큐멘트와 함께 실제 폰트를 보내는 방식을 이용하므로 힌팅(hinting : 서체 디자이너들이 고퀄리티 디스플레이와 출력을 보장할 수 있도록 폰트를 만드는 기능)을 비롯한 다른 캐릭터 정보가 그대로 출력된다. 그러나 폰트 임베딩은 기술적으로 예민한 문제들을 가지고 있다. 파일크기, 전송속도, 스크린 퀄리티는 웹의 하이퀄리티(high-quality)를 보장하기 위해 풀어야 하는 3대 주요 문제로써 도큐멘트를 출력하려는 많은 웹 사이트 방문객들의 변함없는 소망이기도 하다. 특히 배너광고나 헤드라인의 경우 웹에서 제공하는 일반 글꼴 형태가 아닌 변형된 글꼴형태로 출력하기 위해서는 반드시 이미지 에디팅 프로그램을 거쳐서 작업해야 하는 번거로움이 있다. 또한 이미지 데이터는 용량 또한 커서 웹페이지에서 속도부분에서 많은 부하를 가지고 있다. 때문에 웹에서 작은 용량을 가지고, 로딩시 속도도 빠르며, 큰 수정없이 글꼴에 다양한 효과를 낼 수 있는 방법이 필요하다.

Ⅱ. 글꼴 표현 방식

2.1. 글꼴 표현방식에 의한 분류

글꼴은 흔히 외래어로 말해 폰트(font)라고 한다. 사용자의 입장에서 이 폰트는 중요한 것이 된다. 글자의 모양이 사용자와 컴퓨터가 얼마나 친해질 수 있는가를 결정할 수도 있기 때문이다. 폰트의 표현방식에는 여러 가지가 있지만 크게 비트맵(bitmap)방식과 윤곽선(out-line)방식 등이 있다[1].

2.1.1 비트맵 방식

폰트형식중 가장 단순한 형태로 dot의 있고 없음으로 정의되는 폰트이다. 글자의 모양을 일정크기의 점행렬로 표현한 후, 글자에 해당하는 부분은 1, 글자 외부는 0으로 나타낸다. 다시 말하면, 이진 데이터로 이루어진 글자모양이다. 장점은 비디오 메모리에 직접 쓸 수 있기 때문에 속도가 빠르다는 것이다. 작은 글자를 표현할 때 가장 선명하게 출력되고 해상도가 낮은 출력기에서 많이 사용된다.

비트맵 방식의 가장 큰 단점은 기억용량이 크다는 것과 확대할 경우 점의 크기도 같이 커져 계단 현상이 뚜렷이 나타난다. 이런 단점을 보완해서 글자를 크게 확대시켰을때 나타나는 계단 현상을 줄여 부드럽게 표현할 수 있는 것이 있는데 바로 그레이 스케일 폰트라는 것이다.

2.1.2 윤곽선 방식

벡터 데이터의 특징상 프린터를 이용한 출력물은 아주 미려하지만 해 글자의 윤곽선을 직선 또는 곡선으로 나타내고, 그 직선이나 곡선을 생성할 수 있는 제어점으로 폰트를 표현하는 방식이다. 곡선으로 많이 쓰이는 것은 원호(arc), 베지어(bezier), 스플라인(spline) 곡선 등이 있다. 직선은 시작과 종점으로 구성된다. 직선을 그릴 때 시점을 한번 주고 그 후 계속 종점만 알면 선을 그릴 수 있기 때문에 데이터의 양이 적어지는 이점이 있다. 원호는 중심의 좌표와 반지름 그리고 북쪽 방향으로부터의 시작 각도와 끝 각도로 표시할 수 있다.

베지어 곡선과 스플라인 곡선에 대해서는 좀 더 자세히 살펴보면, 이 두 가지 곡선의 특징은 시작과 끝점을 지나며, 곡선의 모양을 결정하기 위해 끌어당기는 컨트롤 포인트가 있어서 곡률을 결정해 준다. 쉽게 말하자면 고무줄의 양 끝을 잡고서 늘리는 것과 같다. 베지어 곡선의 문제점은 지역적 제어가 되지 않아 폰트제작 과정에서의 걸림돌이 되는 것이다. 이런 단점이 보완된 것이 스플라인 곡선인데 속도가 너무 느린 것이 단점이다. 윤곽선 폰트의 장점은 크기에 무관하고 글자모양에 많은 효과를 낼 수 있다. 그러나 글자 크기가 작아지면 획이 사라지고, 끊기고, 겹치는 등의 여러 부작용이 나타난다. 이것을 해결하기 위해서는 힌팅이라는 기술이 필요하다. 힌팅은 작은 글자에서 획의 굵기를 일정하게 조정하거나 획 사이의 간격을 조절하는 기술이다. 윤곽선 폰트는 속도가 느리다는 단점이 있다[2][3]. 윤곽선 폰트를 사용하려면 직선과 곡선을 먼저 그리고 속을 채우는 과정을 거쳐야 하기 때문에 비트맵 폰트와는 비교할 수 없을 정도로 속도가 느리다.

Ⅲ. 웹에서의 글꼴 표현방법

웹에서의 글꼴 표현 방법은 HTML spac 4.01을 보면 자세히 알 수 있다. 실제 운영체제에서 글꼴을 출력하는 방법과 아주 유사하며, 글꼴을 추가로 다운받을 수도 있어서 그 확장성은 무궁무진하다고 할 수 있다. 실제로 HTML에서는 간단한 Tag의 조작으로 폰트 종류, 크기를 설정할 수 있고, 약간의 효과도 줄 수 있게 되어있다. 글꼴에 줄 수 있는 효과들은 Bold, Italic, Stkike, 타자기꼴, 밑줄선, 위첨자, 아래첨자, 현재 글자 크기보다 1 포인트 크게, 현재 글자 크기보다 1 포인트 작게 등의 효과가 있다. 글꼴의 표현은 윈도우즈나 매킨토시의 경우에는 TrueType 글꼴을 쓰고 있으며, TrueType 글꼴은 외곽선 글꼴 형태를 가진 유연성이 좋은 글꼴이다. 때문에 현재 웹페이지에서 Tag의 형식으로 주어지고 있는 효과 이외에도 TrueType 글꼴의 장점을 살려, 벡터 데이터의 성질을 십분 살린 효과도 가능하다.

Ⅳ. 다양한 글꼴 출력을 위한 글꼴 벡터라이징

웹 문서에서는 HTML 내에 글꼴의 고유값을 읽어서 웹 브라우저에 출력하는 방법을 택하고 있다. 때문에 이종 플랫폼간에서 서로의 데이터 호환이 없는 경우나, 현재 가지고 있지 않은 서체로 되어 있는 데이터로 작성된 웹 페이지에서는 글꼴의 깨어지는 현상이 발생한다[1]. 이것은 한글 운영체제를 사용하고 있는 경우, 일본어 시스템에 있는 웹 페이지에 있는 데이터를 읽으려고 할 때 데이터가 깨어지는 현상이 발생하는 것과 동일하다. 이런 문제를 해결하기 위해서는 글꼴을 모두 비트맵 데이터로 변환하면 가능해지는데, 속도나 데이터 용량 문제를 비추어 볼 때 그리 바람직한 방법이 아니다. 또한 웹 브라우저에서 데이터를 표시하는 방법이 텍스트 전용이면 글꼴이 보이지 않는 경우도 발생한다.

때문에 다중 플랫폼에 제한이 없는 글꼴출력을 위해서 웹 브라우저는 외곽선 글꼴의 중요데이터만을 저장하고 있는 형태를 취해야 한다. 폰트 임베딩 방식이 폰트의 정보를 웹 페이지가 가지고 있는 형태라고 한다면, 본 논문에서 제안하는 방법은 글꼴의

외곽선 데이터를 웹 페이지가 가지고 있는 형태를 취한다.

다음은 본 논문에서 제안되어진 고품질 출력을 위한 글꼴 벡터라이징 순서도이다.

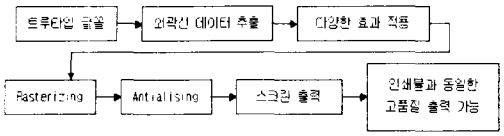

그림 1. 글꼴 벡터라이징 순서
Fig 1. character vectrising sequence

4.1 TrueType 글꼴의 외곽선 추출

트루타입에서의 베지어 커브는 4각의 B-Splines 형태로 되어있다. 각각의 Spline 데이터는 2차 방정식 Beizier의 커브의 연속된 형태로 되어있다. 각각은 3개의 아웃라인 Control Points를 가지고 있다. 만약 각각의 Bezier 커브가 (Ax, Ay), (Bx, By), (Cx, Cy)의 3개의 포인터를 가지고 있으면 Bezier 커브에서의 각각의 포인터의 점은 다음의 공식으로 구할수 있다.

$$Px = (1-t)^2 Ax + 2t(1-t)Bx + t^2 Cx$$
$$Py = (1-t)^2 Ay + 2t(1-t)By + t^2 Cy$$

베지어 곡선에 기반을 둔 벡터 데이터로 글꼴 데이터가 변환되면 사용자의 의도대로 다양한 형태의 변환이 가능하다. 또한 벡터화된 데이터는 데이터양이 다른 포맷에 비해 현저히 적다.

Truetype 글꼴의 아웃라인에 관한 정보는 아래의 TTPOLYCURVE 구조체와 실제의 데이터를 가지고 있는 POINTFX 의 형태로 표현되어질수 있다.

표 1. TTPOLYCURVE 구조체
Table 1. TTPOLYCURVE structure

```
typedef struct tagTTPOLYCURVE { // ttpc
        WORD    wType;
        WORD    cpfx;
        POINTFX apfx[1];
} TTPOLYCURVE, FAR* LPTTPOLYCURVE;
```

POINTFX의 구조체는 Truetype 글꼴에서 글꼴의 아웃라인 데이터를 기술한 점의 실제 좌표값을 포함한다.

표 2. POINTFX의 구조체
Table 2. POINTFX structure

```
typedef struct tagPOINTFX { // ptfx
        FIXED x;
        FIXED y;
} POINTFX;
```

POINTFX 구조체에 대한 설명은 다음과 같다.

- x : TryeType 글꼴의 아웃라인 데이터의 X 값
- y : TryeType 글꼴의 아웃라인 데이터의 Y 값

외곽선이 추출된 트루타입 글꼴 형태는 다음과 같다.

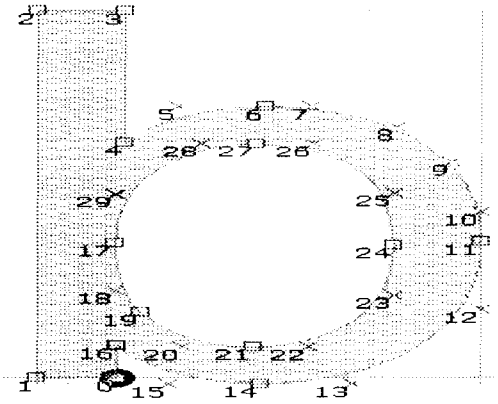

그림 2. 외곽선이 추출된 트루타입 글꼴
Fig 2. true type character extracted outline

V. 다양한 효과 구현

웹상에서 화면에 출력되어지는 글꼴의 형태는 글꼴의 정렬, 크기 조절등만이 HTML 문서에서 직접 제어가 가능하도록 되어있다. 때문에 글꼴에 다양한 형태의 효과를 구현하기 위해서는 여러 절차를 거쳐서 비트맵 데이터로 만들어야 하는 불편함을 수반하게 된다. TrueType 글꼴은 외곽선 글꼴의 형태이므

로 여러 가지 효과를 주는 것이 가능하다. 때문에 특정 부분의 강조나 헤드라인 글꼴에 많이 쓰이는 기능을 미리 HTML Tag로 정의하거나, Java의 Plug-in 형태로 제공하게 된다면, 웹 디자이너 들은 많은 노력을 들이지 않고 더욱 다양한 효과를 낼 수 있게 된다. 본 논문은 문자꼴 변형 형태로써 많이 쓰이는 파노라마 효과와 글꼴의 외곽선 패턴 분할의 예를 대표적으로 든다.

5.1 파노라마 효과

파노라마 효과는 글꼴 전체적인 모양을 정해진 곡선이나 직선의 패턴으로 바꿔주는 현상을 일컫는다. 파노라마 효과의 예를 들어보면 다음과 같다.

파노라마 효과 원리는 주어진 파노라마 곡선 패턴의 형태로 TrueType 글꼴의 데이터를 이동시켜줌으로써 간단히 구현할 수 있다. 본 논문에서는 파노라마의 형태에 따라서 20가지로 분류하였다.

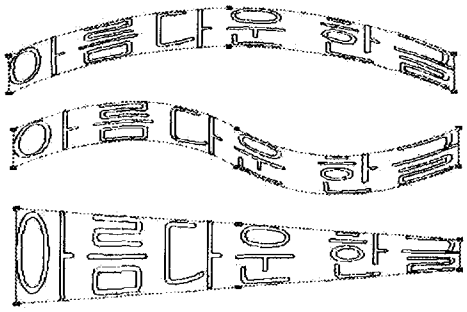

그림 3. 파노라마효과가 적용된 예
Fig 3. applied example of panorama effect

그림 4. 모양에 따른 분류
Fig 4. classification as form

5.2 글꼴 아웃라인의 임의 패턴 분할

일단 글꼴의 외곽선 데이터를 패턴분할하기 위해서 직선일때와 곡선일때를 나누어 처리한다. 글꼴의 외곽선 데이터의 값을 구한 후 직선일때는 Brensenham 알고리즘을 이용해서 임의의 길이의 패턴분할을 하였고, 곡선에서는 Casteljar 알고리즘을 바탕으로한 선형 보간법을 이용한 재귀호출 방법을 사용하여 임의의 패턴분할을 행하였다. 예를 들면 히읏(ㅎ)의 경우에서 위의 두 획은 직선에서의 패턴분할 방법을 사용하였고, 아래의 이응(ㅇ)은 곡선에서의 패턴분할을 이용하였다.

Truetype 글꼴의 외곽선은 벡터데이터로 되어 있기에 본 논문에서는 벡터데이터의 임의 패턴분할에 중점을 둔다. 일단 글꼴의 외곽선 데이터를 베지어 곡선형태로 변환 후 사용하기로 한다. 베지어 데이터는 4개의 점으로 이루어지며, 2개의 양 끝점 데이터와, 그 두 점의 방향성과 길이 데이터를 가지고 있는 2개의 컨트롤 포인트로 이루어져있는데, 이 두 개의 컨트롤 포인트를 이용해서 쉽게 곡선을 수정할 수 있다는 장점을 가지고 있다. 때문에 본 논문에서는 베지어 커브에서의 임의의 패턴분할을 어떻게 행할 것인가에 대해서 중점적으로 다루어진다. 또한 임의의 패턴분할을 위해서 주어지는 패턴은 unsigned integers 두 개가 한 쌍으로 구성되어 지며, 각 숫자가 라인조각의 길이를 가리키고 있고, 두번째 숫자는 다음 라인까지의 gap을 나타내도록 설계했다. 단 패턴은 효율성 측면에서 6개로 제한한다. 예를 들면 dash-dot-dot 패턴을 그리기 위해서는 아래와 같은 패턴이 요구되어지고, 본 논문에서 제시되는 패턴의 형태는 다음의 형태를 따른다. 6개의 숫자는 Dash, gap, dot, gap, dot, gap 의 형태를 따른다.

```
unsigned Pattern[] = { 30, 10, 10, 10, 10, 10 };
```

그림 5. 임의의 패턴
Fig 5. unsigned pattern

5.2.1 Line에서의 패턴 분할

라인에서의 패턴분할은 먼저 직선을 효율적으로 임의의 길이의 패턴으로 나눠야 한다. 또한, 두 점사이의 직선을 그릴 때 패턴에 맞추어서 그려야 하기 때문에 속도의 효율성면에서, 직선 그리는 알고리즘으로 가장 많이 쓰이고 있는 Brensenham 알고리즘을 사용해서 두 점사이를 보간해 가면서 일정한 패턴으로 분할을 행한다.

직선에서의 패턴분할을 위한 기본 알고리즘은 다

음과 같다

step 1. 패턴분할할 양 끝점을 선정한다.

step 2. 임의의 길이의 패턴을 정의한다. 패턴은 서론에서 정의한 형식을 따른다.

step 3. moveto()와 lineto() 함수를 적절히 이용해서 정의된 패턴을 하나씩 증가시키면서 순차적으로 직선을 분할한다. 이때 lineto 함수는 내부적으로 brensenham 알고리즘을 사용해서 보간한다.

step 4. 모든 패턴길이의 합이 두 끝점의 길이의 합보다 크면 종료한다

다음은 직선에서의 패턴 분할 예이다.

그림 6. 직선에서의 패턴 분할의 예
Fig 6. example of pattern division on the line

5.2.2 Curve 패턴 분할

곡선에서의 패턴분할은 베지어 커브의 패턴분할을 통해서 이루어진다. 베지어 커브는 양끝 2점과 2개의 제어점으로 이루어지는데, 베지어 곡선을 임의의 길이의 패턴으로 나누기 위해서는 먼저 베지어 곡선의 길이를 구해야 한다.

제어점 Q_0, Q_1, ..., Q_n을 가진 n차 Bezier 곡선 b(t)에서 곡선의 길이는 아래의 수식에 의해 구해진다[1, 2].

$$\text{arc length} : L(b) = \int_1^0 b'(t)\,dt,$$

$$\text{polygon length} : Lp(b) = \sum_{i=1}^{n} |Qi - Qi-1|,$$

$$\text{chord length} : Lc(b) = |Qn - Q0|,$$

Casteljar's algorithm을 통해서 특정점에서의 곡선의 분할이 가능하다. 곡선의 분할은 아래의 수식으로 가능하다.

$$Q_i^0 = Qi, \qquad i=0,...,n,$$

$$Q_i^0 = \frac{1}{2} Q_i^{k-1} + \frac{1}{2} Q_{i+1}^{k-1},$$

$i=0,...,n-k,\ k=1,...,n$

곡선에서는 Casteljau 알고리즘을 이용한 선형 보간법에 의한 재귀호출 방법으로 직선처럼 충분히 짧은 여러 개의 선으로 베지어 커브를 나눌 수 있다. casteljaur 알고리즘은 베지어 커브의 제어점을 연결하는 다각형의 길이의 합과, 분할하려고 하는 길이의 차가, 주어진 오차범위(보통 0.05)내에 들때까지 재귀호출함으로써 베지어 커브를 분할하는 알고리즘이다. 그것은 베지어 커브를 dash나 dot로 구성되어진 여러 개의 짧은 곡선 조각으로 나누는 것도 역시 가능하게 한다. 3차 곡선 형태로 구성되어진 베지어 커브는 0과 1사이의 t 값으로 특징적인 표현이 가능하게 하며, 베지어 커브는 곡선 위 어느 점에서나 특징적인 t 값을 가지고 나눌 수 있다. 선형보간 후에 파라미터 t 값은 2등분되며, 이것은 베지어 커브 안에서 우리가 임의의 길이를 얻을 때 t 값을 사용할 수 있다는 것 또한 알 수 있다.

다음은 베지어 커브에서의 적절한 arc의 길이를 구하는 프로그램이다.

```
float BezierLength(float b[4],float eps)
{
        Lp = poly_length(b);
        Lc = chord_length(b);
        n = degree(b);
        err = error();
        if(err<eps)
        return (2*Lc+(n-1)*Lp)/(n+1);
        else
        {
        b1, b2 = subdivision(b);
        eps1,eps2 = tolerance();
        return length(b1,eps1)+length(b2,eps2);
        }
}
```

곡선에서의 패턴분할을 위한 기본 알고리즘은 다음과 같다

step 1. 패턴분할 할 베지어 커브를 선정한다.

step 2. 임의의 길이의 패턴을 정의한다. 패턴은 서론에서 정의한 형식을 따른다.

step 3. 베지어 커브를 임의의 길이로 분할하기 위해 주어진 베지어 커브를 베지어 곡선을 나누기 위한 가장 효율적인 알고리즘인 casteljaur algorithm으로 분할한다.

step 4. 모든 패턴길이의 합이 두 끝점의 길이의

합보다 크면 종료한다.

Ⅵ. 래스터라이징

아웃라인 되어진 벡터 데이터를 스크린이나 프린터에서 고품질 출력을 위해서는 데이터를 비트맵화하는 과정이 필요하다. 이러한 과정을 래스터라이징 이라고 한다. 래스터라이징에서는 다음과 같은 과정이 수반된다

-폰트 데이터를 읽어서 해석
-확대, 축소를 위해서 제어점의 좌표를 계산
-회전, 기울임 등의 변형 처리
-힌팅 처리
-윤곽선의 내부를 채움(Scan conversion)
-필요할 경우 역상, 밑줄 등을 처리

힌팅은 작은 글자에서 획의 굵기를 일정하게 조정하거나 획 사이의 간격을 조절하는기술이다.

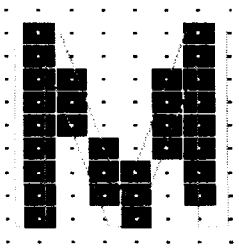

그림 7. 힌팅되기 전
Fig 7. before hinting

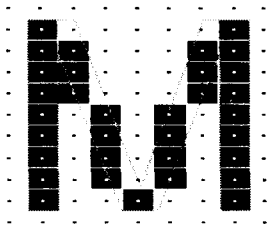

그림 8. 힌팅 된 후
Fig 8. after hinting

래스터라이징에서 Scan-Conversion은 폰트의 아웃라인 데이터에 Coloring 하는 작업을 말하며, 다음과 같은 과정을 통해서 이루어진다.

Step 1. Measurement : 글꼴의 아웃라인 정보를 pixel 또는 contour 단위로 스캔, Stpe 2 나 Step 3 에서 필요한 메모리 계산
Step 2. Rendering : 스캔되어진 데이터의 Line과 Spline의 교차점을 구하여 찾아진 교차점들은 왼쪽에서 오른쪽으로 정렬
Step 3. Filling : 정렬된 교차점들로 비트맵을 위에서 아래쪽으로 픽셀을 셋팅
Step 4. Dropout control : Dropout이 가능하면, 재스캔후 픽셀을 셋팅

그림 9는 Scan Conversion의 예이고, 그림 10은 Dropout된 후 변경된 글꼴 데이터를 보여준다.

그림 9. 스캔 변환
Fig 9. Scan Conversion

그림 10. 글꼴 아웃라인, Dropout 전, Dropout 후
Fig 10. character outline, before dropout,after dropout

Ⅶ. 앤티에일리어싱

스크린은 픽셀 장치에 종속적으로써 다각형이나 커브 출력시 수평, 수직 라인에서 계단 현상이 발생한다. 때문에 계단 현상을 줄이기 위해서 픽셀에 그림자 효과를 적용시켜 계단 현상을 감소시키는 기법이 필요하게 되는게 이러한 기법은 앤티에일리어싱(Anti-Alising) 이라고 한다. 글꼴의 계단현상 제거 방법으로는 "Sampling Filters"를 이용한 방법이 주로 이용된다.

그림 11. 일반도형과 Antialised 되어진 도형
Fig 11. general figure and antialised figure

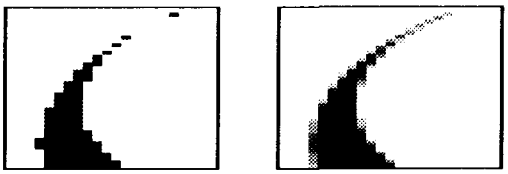

그림 12. 2배 확대의 예
Fig 12. example of double Magnification

Ⅷ. 웹에서의 적용

일반적으로 HTML 문서에서 글자의 속성을 표현하는 방법은 다음의 형태를 따른다.

1. 굵은 글씨로 나타내기(Bold) <B> ... </B>
2. 이텔릭체로 나타내기(Italic) <I> ... </I>
3. 타자기 활자체로 나타내기(Teletype) <TT> ... </TT>
4. 인용(Citation) <CITE> .. </CITE>
5. 코드(Code) <CODE> ... </CODE>
6. 강조(Emphasis) <EM> ... </EM>
7. 키보드(Keyboard) <KBD> ... </KBD>
8. 변수(Variable) <VAR> ... </VAR>

글자 속성을 지정하는 <B>, <I>, <TT> 같은 태그들은 표현되어야 할 문자의 속성을 구체적, 직접적으로 지정해주며 웹 브라우저 종류에 상관없이 태그가 의미하는 형태로 보여주므로 이러한 태그를 물리적 유형(Physical Style) 태그라고 한다. 이와 반대되는 개념으로 태그에 대한 기능으로서 글자 속성을 분류하는 것들이 있는데 이를 논리적 유형(Logical Style) 태그라고 한다. 인용, 코드, 강조, 키보드, 표본, 강조된 활자 변수 같은 태그 등은 브라우저로 하여금 그 성격에 맞게 처리하도록 하기 때문에 동일한 명령어라도 브라우저에 따라서 표현되는 외형적 양식이 조금씩 다를 수도 있고, 서로 다른 명령어들도 동일한 표현형식을 취할 수 있다.

실제로 제안되어진 글꼴의 효과를 웹페이지에 적용하기 위해서는 태그에 적용시킨다면 물리적 태그를 이용할 수 있다.

파노라마 형태의 태그는 다음과 같은 형태로 정의해 볼수 있다.

```
<PANORAMA style-3 offset-50 brushcolor-white
    pencolor-black penwidth-1> 아름다운 한글
</PANORAMA>
```

또한 임의의 패턴분할 형태는 다음과 같은 형식으로 제안해 볼 수 있다.

```
<PATTERN pattern-50,5,15,5,15,5 brushcol-white
pencolor-black penwidth-1>아름다운 한글
</PATTERN>
```

Ⅸ. 구현 및 분석

실제적인 구현은 웹에서 직접 파노라마 효과나 외곽선의 패턴 분할 효과를 갖도록 하는 것으로 다음과 같이 고려해 볼 수 있다.

첫째, HTML 문서에 Tag 형태로 첨가하여 기존의 글꼴 출력형식과 제안되어진 형식을 접목시키는 방법

둘째, Active X, PulgIn 및 Applet 형태로 구현하는 방법

첫째 방법이 둘째 방법에 비해서 훨씬 바람직하지만, 구현이 쉽고 간편한 두 번째 방법을 선택하였다. 프로그래밍 언어는 차후 Active X나 PlugIn 이 용이한 Visual C++ 6.0을 사용하였다.

● 글꼴에 파노라마 효과를 준 경우

sample 1. <PANORAMA style=9 offset=50 brushcolor=white pencolor=black penwidth=4>아름다운 한글</PANORAMA>

그림 13. PANORAMA style=9, offset =50
Fig 13. PANORAMA style=9, offset =50

sample 2. <PANORAMA style=6 offset=20 brushcolor=white pencolor=black penwidth=4>아름다운 한글</PANORAMA>

그림 14. PANORAMA style=3, offset=20
Fig 14. PANORAMA style=3, offset=20

sample 3. <PANORAMA style=8 offset=60 brushcolor = white pencolor = black penwidth = 4>아름다운 한글</PANORAMA>

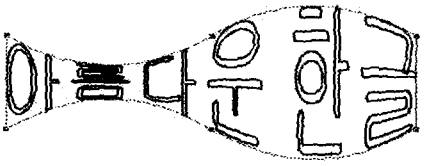

그림 15. PANORAMA style=8 offset=60
Fig 15. PANORAMA style=8 offset=60

● 글꼴 외곽선에 임의의 패턴분할을 한 경우

sample 1. PATTERN pattern= 20,5,20,5,20,5 brushcol=white pencolor=black penwidth=1> 아름다운 한글</PATTERN>

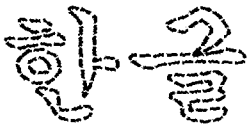

그림 16. Pattern = {20,5,20,5,20,5}
Fig 16. Pattern = {20,5,20,5,20,5}

sample 2. PATTERNpattern=60,10,20,10,20,10 brushcol=white pencolor=black penwidth=1> 아름다운 한글</PATTERN>

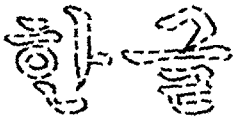

그림 17. pattern = {60,10,20,10,20,10}
Fig 17. pattern = {60,10,20,10,20,10}

sample 3. PATTERN pattern=40,40,40,40,40,40 brushcol=white pencolor=black penwidth=1> 아름다운 한글</PATTERN>

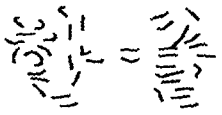

그림 18. pattern = {40,40,40,40,40,40}
Fig 18. pattern = {40,40,40,40,40,40}

sample 4. PATTERN pattern= 40,10,20,40,10,20 brushcol=white pencolor=black penwidth=1> 아름다운 한글</PATTERN>

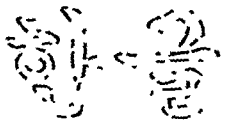

그림 19. pattern = {40,10,20,40,10,20}
Fig 19. pattern = {40,10,20,40,10,20}

아래 그림은 Java로 웹상에서 구현된 예이다.

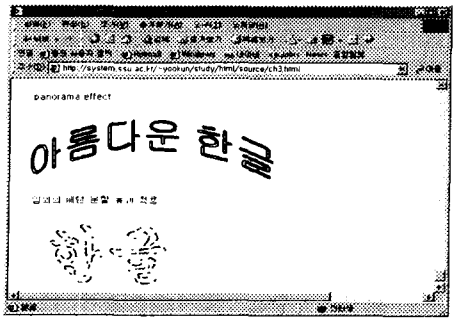

그림 20. Java로 웹상에서 구현
Fig 20. implementation on the web by java

X. 결 론

벡터 데이터로 변환되어진 글꼴 데이터는 사용자가 이용하기에 따라서 여러 분야로의 변형이 가능하다. 변환되어진 벡터 데이터는 용량이 작고, 쉽게 변형이 가능하여 웹상에서 글꼴을 다양하게 표현할 수 있다.

Truetype 글꼴에서 아웃라인 데이터를 이용한 파노라마 기능 구현은, 베지어(Bezier) 데이터로 변환되어진 글꼴의 아웃라인 데이터의 적절한 변형으로 가능하였고, 임의의 길이의 패턴분할은 베지어 커브의 길이 분할을 이용해서 가능할 수 있었다.

특히, 본 논문에서 기대되는 효과는 웹에서 플랫폼에 영향을 받지 않고 글꼴 출력 이 가능하며, 파노라마 효과나, 외곽선 패턴 분할 효과를 간단한 태그를 사용하여 표시할 수 있다. 또한 웹 페이지 제작시 벡터데이터로 변형해서 할 수 있는 작업들을 쉽게 표현할 수 있으며, 래스터라이징(Rasterizing)과 벡터라이징(Vectorizing)을 거친 글꼴 출력은 다른 그래픽 에디터를 통하지 않고도 헤드라인이나 베너에서 높은 품질을 가진 효과를 낼 수 있다.

그러나, 임의의 패턴분할 과정시 각 직선이 예각으로 만나는 경우 두 선분이 매끄럽게 연결되지 않는 현상이 발생할 수 있고, 각 글꼴에서 자음과 모음이 서로 겹치는 부분의 경우 데이터를 보정해야 하는 문제가 발생하였다. 또한 확대 축소시 소숫점 연산 과정에서 생기는 오차가 발생하여 원래의 벡터 이미지가 변형되는 현상이 발생하였다.

향후 연구과제로는 커브의 만나는 점에서의 매끄러운 처리방법과 화면 확대 축소시에 발생하는 이미지 왜곡현상 개선 및 실제 HTML문서 형식의 Tag에 기능을 추가해서 타당성을 검토해 보는 것도 필요하리라 본다.

참 고 문 헌

[1] Guenter and R.Parent. "Computing the arc length of parametric curve". IEEE Computer Graphics and Applications, pp 72-78, Mar. 990.

[2] Jens Gravesen. "Adaptive subdivision and the length of Bezier curves" mat-report no. Mathematical Institute, The Technical University of Denmark, Nov. 1992,

[3] T. Lindgren, J. Sanchez, D.Kirk, and J. Hall. "Curve tesselation criteria through sampling". Graphics Gems III, Academic Press, pp 262-265, 1992

[4] Roger D. Hersch, Ecole Polytechnique Federale, and Lausanne, "Visual and Technical Aspect of Type", Cambridge University, Press, pp.110-125, 1993.

저 자 소 개

김용호(Yong-Ho Kim)
1989년 광주대학교 전자계산과 졸업(공학사)
1993년 경남대학교 전자계산과 대학원(공학석사)
1999년~현재. 조선대학교 전자계산학과 박사과정
※관심분야 : 인공지능, 전문가시스템, 멀티미디어, 데이터베이스, 정보보안, 전자상거래

이윤배(Yun-Bae Lee)
1980년 광운대학교 전자계산학과 졸업(이학사)
1983년 광운대학교 대학원 전자계산학과 졸업(이학석사)
1993년 숭실대학교 대학원 전자계산학과 졸업(공학박사)
1988년 4월~현재 조선대학교 컴퓨터공학부 교수
1999년 7월~2000년 현재 광주광역시 시정정책자문회의 위원
1996년 7월~2000년 현재 광주광역시 및 전라남도 지역 정보화 추진위원
※관심분야 : 인공지능, 전문가시스템, 멀티미디어, 데이터베이스, 정보보안, 바이러스