The Asian Journal of TEX, Volume 4, No. 1, June 2010
Article revision 2010/6/12

Fontspec: XeTeX의 날개 Fontspec: Wing of XeTeX

이주호* Juho Lee
국회예산정책처 latex.juho@gmail.com

Keywords XeTeX, fontspec, font format, Advanced Typography, OpenType, TrueType, OpenType layout features, glyph substituion, glyph variant

Abstract Fontspec 패키지는 XeTeX에서 트루타입 폰트와 오픈타입 폰트를 쉽게 사용할 수 있도록 도와주는 패키지이다. 이를 통하여 사용자는 폰트에 포함된 고급스러운 타이포그래피적인 특성을 자유롭게 사용하여 고품질의 문서 조판을 이루어낼 수 있다. 이 글은 폰트에 대한 이해를 도모하기 위하여 현대 인쇄 출판에 주로 사용되는 폰트의 발전 과정과 각 폰트의 특성을 간략히 소개한다. 이어 리거처, 숫자 모양, 커닝, 작은 대문자, 스와시 · 올터네이트 · 배리언트 등 글리프 대체로 일괄할 수 있는 폰트의 고급 특성을 fontspec 패키지 및 한글 텍 환경에 최적화된 XeTeX-ko에서 어떻게 구현할 수 있는지 다양한 예제 중심으로 제시한다.

1 들어가며

2005년 경 등장한 XeTeX[1]은 텍의 폰트 포맷인 TFM (TEX font metric) 문제로 고민하던 수많은 사람들에게 아주 매력적으로 다가왔다. XeTeX 이전에 ko.TEX의 기본 바탕 글꼴인 은바탕과 굵은 은바탕 두 가지 글꼴을 쓰기 위해 1천 개가 넘는 `.tfm` 파일이 필요했다. `.tfm`만 필요한 게 아니라 `.vf`, `.map`, `.fd` 등 고려해야할 게 많았다. 그러나 XeTeX이 등장하면서 사정은 나아졌다. 사용자는 XeTeX으로 자신의 컴퓨터에 들어있는 트루타입 폰트와 오픈타입 폰트를 쉽게 사용할 수 있다. 예를 들어 XeTeX에서 Cambria 폰트의 굵은 이탤릭체 작은 대문자(bold italic small captial)를 사용하려면 다음과 같이 하면 된다.

```
\font\Cambria="Cambria/BI:+smcp" at 10pt
\Cambria Cambria Bold Italic Fonts
```

CAMBRIA BOLD ITALIC FONTS

그러나 Plain TEX의 방식과 닮은 이 XeTeX의 폰트 지정 방식이 때로는 낯설고 어렵게 느껴지는 사람도 있을 것이다. 또 사용자가 자신의 시스템에 있는 어떤 트루타입 폰트나 오픈타입 폰트를 지정하면서 LaTeX에서 사용하는 `\emph`, `\bfseries`, `\textbf` 등의 폰트 명령을 사용하려면 일일이 정의를 새로 해주어야 한다. 크기와 관련된 명령 `\Large`, `\small` 등을 사용하려면 이에 맞는 보통 모양, 굵은 모양, 이탤릭 등의 정의를 새로 해주어야 했다. 윌

*세밀한 한글 타이포그래피를 구현할 수 있도록 XeTeX-ko를 개발한 김도현 교수께 감사의 뜻을 전한다.

1. XeTeX과 XeLaTeX의 관계는 TEX과 LaTeX의 관계와 같다.

로버트슨(Will Robertson)이 만든 fontspec 패키지는 XeTeX(엄밀히 말하면 XeLaTeX)에서 이러한 번거로움을 해소하기 위하여 만들어진 패키지이다. 이 패키지를 사용하면 사용자는 폰트를 직관적으로 지정할 수 있고 폰트에 내재된 고급 특성도 쉽게 구현할 수 있다.

이 글의 목적은 fontspec 패키지의 여러 가지 사용법을 적절한 예와 함께 설명하는 것이다. 설명하면서 '폰트'와 '글꼴'은 구분하지 않고 사용하려고 한다. 이에 앞서 몇 가지 용어와 배경을 짚고 넘어간다.

1.1 캐릭터와 글리프

베르너 렘버그(Werner Lemberg)는 [21]에서 캐릭터(character)와 글리프(glyph)에 대해 다음과 같이 설명하고 있다.

> *Characters* are entities which have a semantic meaning. Visual presentation forms of characters are called *glyphs*. A character can be represented by more than a single glyph—just think of an italic *A* and a sans-serif A.

이에 덧붙여 [3], [4], [18] [26] 등의 설명에 따르면 캐릭터와 글리프는 우리말로 대략 다음과 같이 정의할 수 있을 것이다.

캐릭터 캐릭터는 어떤 언어의 의미론적 최소 단위이다. 예를 들어 'ㄱ'이라고 했을 때 우리는 '기역'이라고 읽고 한글 자모의 첫 번째 글자라는 의미임을 알고 있다. 또 'a'라고 했을 때 우리는 '에이'라고 읽고 영어 알파벳의 소문자 첫 번째 글자임을 알고 있다.

글리프 글리프는 어떤 폰트에 담겨 있는 캐릭터의 특정한 모양이다. 예를 들어 나눔고딕 글꼴의 가, 서울한강체 M 글꼴의 가의 모양은 각각의 폰트에 담긴 '가'라는 캐릭터의 특정한 모양이다. 또 Palationo Italic의 *a*, Palatino Bold의 **a** 모양은 각각의 폰트에 담긴 'a'라는 캐릭터의 특정한 모양이다.

1.2 폰트의 형식

폰트 안에 담긴 글리프를 그리는 방식은 크게 비트맵(bitmap) 형식과 아웃라인(outline) 형식이 있다. 비트맵 폰트는 일정한 영역 안에 하나의 글리프를 나타낼 때 점을 찍어 표현하는 방식이다. 점을 많이 사용하면 곡선 처리가 부드러워져 자연스럽게 보이고 점을 적게 사용하면 거칠게 보인다. 이에 따라 비트맵 폰트는 처음 용도에 맞게 그려진 크기를 벗어나 확대하면 십중팔구 계단 현상이 일어난다.

아웃라인 폰트는 윤곽선 폰트, 스케일러블(scalable) 폰트, 벡터 폰트로도 부르는데 글리프의 윤곽을 수학적으로 계산된 식—3차 베지어(Bézier) 곡선 또는 2차 B-운형(雲形; spline) 곡선—으로 표현하기 때문에 크기의 확대 및 축소에 영향을 받지 않는다.

이 중 전자출판에서 주로 사용되는 대표적 아웃라인 폰트 형식을 알아본다.

포스트스크립트(*PostScript*) 포스트스크립트 폰트는 어도비(Adobe) 사가 개발한 폰트 형식으로 3차 베지어 곡선을 사용하며 당연히 아웃라인 폰트이다.

타입 *1 (Type 1)* 가장 기본이 되는 포스트스크립트 폰트 형식이다. 1바이트 라틴 문자권을 위한 폰트 형식으로 화면과 출력의 괴리를 줄이는 고급 힌팅(hinting) 정보가 들어 있다. 글리프는 최대 256자까지 담을 수 있다. 마이크로소프트(Microsoft) 윈도에서 `.pfm`은 타입 1 폰트의 폰트 메트릭(metric) 정보를 담고 있는 파일이고 `.pfb`는 폰트의 아웃라인 정보를 담고 있는 파일이다. 타입 1은 인쇄출판계의 사실상 표준(de facto standard)으로 군림해오고 있으며 나중에 타입 3, 타입 0, 타입 2, 멀티플 마스터 형식의 기본이 되었다.

타입 *3 (Type 3)* 힌팅 알고리즘을 빼는 대신 모든 포스트스크립트 언어를 사용할 수 있게 한 폰트로, 타입 1보다 질이 떨어지고 폰트 크기도 크지만 구현할 수 있는 기능이 많다. 역시 256개의 글리프를 보유할 수 있다. 한글 글꼴과 같은 2바이트 코드 용도로 사용되기도 하였다.

타입 *0 (Type 0)* 타입 1이나 타입 3이 가질 수 있는 256개의 글리프만으로는 한국과 중국, 일본 등의 글자를 표현하기에는 턱없이 부족하다. 타입 0은 이러한 문제를 해결하기 위해 타입 1이나 타입 3을 2바이트 폰트로 구성해주는 합성(composite) 폰트이다. 타입 0 구조를 가지는 대표적인 예는 CID-keyed 폰트와 OCF (original composite font)가 있다.

CID-keyed CID-keyed 폰트는 OCF 폰트의 문제점을 극복해 2바이트 문자 체계를 효율적으로 처리하도록 만든 포맷이다. 모든 글리프가 캐릭터 식별번호(CID; character identifier)를 가지고 있고, CMAP (character map) 파일을 이용하여 캐릭터와 글리프 대응을 빠르게 처리할 수 있다. 주로 한국과 중국, 일본의 폰트를 위해 개발되었다.

CFF*/타입 *2 (Type 2) 타입 2는 엄밀히 말하면 폰트의 포맷은 아니다. 타입 2는 타입 1의 *charstring*을 더 강력하고 더 효과적으로 만든 것을 말한다. CFF (compact font format) 또한 폰트의 포맷은 아니다. 이는 아웃라인 폰트의 압축 기술을 의미한다. CFF/타입 2 형식을 담은 오픈타입 폰트를 '타입 1 오픈타입'이라 부른다. Yannis Haralambous는 [18]에서 CFF/타입 2를 가리켜 어도비 폰트 테크놀로지의 '완벽한 예(ne plus ultra)'라 하였다.

타입 *42 (Type 42)* 타입 42는 트루타입 폰트를 포스트스크립트 프린터로 인쇄할 때 사용되는 폰트 형식이다. 타입 42를 사용하면 트루타입 폰트를 포스트스크립트 타입 1이나 타입 3로 변환하여 인쇄하는 것보다 더 나은 출력 품질을 보장할 수 있다.

멀티플 마스터*(Multiple Master)* 포스트스크립트 타입 1에서 발전한 멀티플 마스터(MM) 폰트는 쉽게 말해 마스터 폰트(원본 폰트)를 두 개 이상 정해놓고 그 사이에 있는 폰트는 각각의 마스터로부터 뽑아낸 두께(weight), 자폭(width), 시각적 크기(optical size; 캡션, 본문, 작은 제목, 큰 제목 크기 등) 정보로 보간(interpolation)하여 구현하는 방식이다. 예를 들어 A라는 마스터 폰트는 굵기

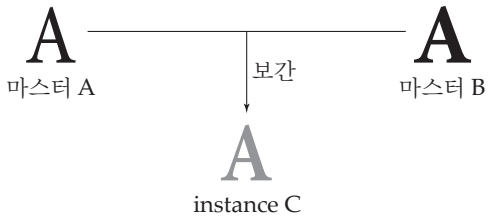


그림 1. 멀티플 마스터 폰트의 원리

10, 자폭 80, 크기 30이고 B라는 마스터 폰트는 굵기 20, 자폭 40, 크기 60이라고 가정하자. 그러면 마스터 A와 마스터 B의 정보를 이용하여 그 중간의 느낌을 갖는 굵기 15, 자폭 60, 크기 45도 정도의 C 폰트를 만든다는 원리이다. 이렇게 보간하여 만든 C 폰트를 이쪽 용어로 instance라 한다. (그림 1 참조)

그러나 멀티플 마스터 폰트를 지원하는 유틸리티와 장비가 적어 별로 인기를 끌지 못했다. [23]에 의하면 2004년 기준으로 약 50종 미만의 멀티플 마스터 폰트가 제작되었으며 그중의 절반 이상이 어도비 사가 만든 것이라고 한다.

트루타입 *(TrueType)* 트루타입 폰트는 애플(Apple) 사와 마이크로소프트 사에서 공동으로 만든 아웃라인 폰트 형식으로 마이크로소프트 윈도와 맥 OS에서 기본으로 제공된다. 원래 이름은 트루이미지(TrueImage)였는데 나중에 트루타입으로 바뀌었다. 트루타입은 아웃라인을 구현하기 위하여 2차 B-운형 곡선을 이용하는 것, 바깥쪽 윤곽선(contour)의 진행방향은 시계방향, 안쪽 윤곽선의 진행방향은 반시계방향인 것에 비해 포스트스크립트 타입1은 3차 베지어 곡선을 이용하고 윤곽선의 진행방향은 트루타입과 반대인 차이점을 가지고 있다.

트루타입은 테이블 구조로 이루어져 있고 글자폭 등을 나타내는 매트릭스 정보, 폰트 이름, 힌팅 데이터 등을 담고 있으며 포스트스크립트보다 뛰어난 힌팅 정보를 담을 수 있는 특징이 있다. 또 기본적으로 2바이트 코드를 지원하여 한국과 중국, 일본 등에서 빠르게 수용되었다. 확장자 `.ttf` 외에도 여러 `.ttf`를 합쳐놓은 `.ttc`도 있다.[2]

애플 사는 이 트루타입 기술을 맥 OS에서만 작동하는 트루타입 GX로 발전시켰는데 나중에 Mac OS X 운영체제의 핵심적 폰트 구현 방식인 AAT (Apple Advanced Typography)로 진화하게 된다.

마이크로소프트 사는 어도비 사와 공동으로 트루타입 기술을 발전시킨 트루타입 오픈(TrueType Open)을 개발하게 된다. 어도비 사는 포스트스크립트 폰트와 트루타입 폰트의 간극을 메우기 위해 자사의 포스트스크립트 기술의 하나인 CFF를 트루타입 폰트에서 해석할 수 있도록 도와줌에 따라 트루타입 오픈은 한층 더 똑똑해졌다. 트루타입 오픈은 나중에 오픈타입으로 이름이 바뀌게 된다.

오픈타입 *(OpenType)* 오픈타입 폰트는 마이크로소프트 사와 어도비 사가 함께 만든 유니

2. 마이크로소프트에서 제공하는 TTSDK 안에 들어있는 BREAKTTC 유틸리티를 이용하면 `.ttc`를 여러 개의 `.ttf`로 쪼갤 수 있다.

코드 기반의 아웃라인 폰트 포맷이다. 오픈타입 폰트는 트루타입 폰트의 특성과 포스트스크립트 폰트, 특히 CFF의 특성을 합쳐놓은 것이라고 생각하면 된다. 과거에 특정 폰트가 특정 운영체제에 종속되었던 것에 비해 오픈타입 폰트는 윈도, 맥, 리눅스 등 운영체제를 가리지 않고 무리 없이 사용할 수 있다. 오픈타입 폰트의 확장자는 `.otf` 또는 `.ttf`이다. 확장자가 `.otf`인 것은 포스트스크립트 폰트를 기반으로 한 것으로 CFF 아웃라인 형식을 담은 타입 1 오픈타입을 의미하고, 확장자가 `.ttf`인 것은 트루타입 폰트를 기반으로 한 것으로 트루타입 아웃라인 형식에 오픈타입 정보가 들어있다.

어도비 사에서 포스트스크립트 폰트를 화면용이나 인쇄용으로 사용할 수 있도록 ATM (Adobe Type Manager) 이란 소프트웨어를 배포했다. 마이크로소프트 윈도의 경우 98 버전 이하에서 포스트스크립트 폰트를 사용하려면 화면용으로든 인쇄용으로든 ATM이 필요했다. 맥의 경우 OS 9 버전 이하에서 인쇄용으로 별도의 조치없이 포스트스크립트 폰트를 사용할 수 있었는데 화면에 띄우려면 역시 ATM이 필요했다. 그러나 마이크로소프트 윈도 2000, XP 버전 이상, Mac OS X 버전부터 운영체제 자체에 포스트스크립트 타입 1이나 타입 1 오픈타입의 해석 능력을 보유하게 되어 ATM이 필요하지 않게 되었다.

결국 포스트스크립트 폰트는 오픈타입 폰트라는 더 향상된 폰트 형식으로 종착한 셈이다. 많은 운영체제에서 오픈타입을 지원하게 됨에 따라 이들 폰트의 종류가 많이 늘어났다. 인터넷의 발달과 더불어 오픈타입 폰트가 각광받는 것은 당연지사라 할 수 있다.

그러나 맹목적으로 오픈타입 폰트를 사용하는 것은 한번쯤 생각해볼 문제이다. 오프셋 인쇄를 하기 위해 반드시 거쳐야할 필름 출력소에서는 여전히 포스트스크립트 폰트를 많이 사용하고 있다. 힌팅은 예전에 화면과 출력물의 괴리를 최소화하기 위한 기술이었으나 요즘은 스크린 해상도와 디스플레이 기술이 많이 발전하여 힌팅의 의미가 퇴색되고 있다. 이 때문에 트루타입의 더 나은 힌팅 기술을 이용하기 위하여 포스트스크립트 대신 사용한다는 말도 구시대의 얘기가 되어 버렸다.

한편 포스트스크립트 타입 1 폰트를 이런저런 폰트 에디터에서 트루타입이나 오픈타입으로 변환하면서 아웃라인 정보가 달라지거나[3] 애초에 담겨있던 고급스런 힌팅 정보가 사라지고, 기본 라틴 캐릭터 외에 다른 언어의 캐릭터 세트는 유실되는 등 오히려 품질이 조악해지는 경우도 있다. 또 타입 1 시절에는 폰트 패밀리를 유지했던 폰트들이 낱개로 분리되는 경우도 있어 문서 조판에서 폰트의 완벽한 일체감을 이루지 못하는 경우도 있다. ([3], [4], [8], [9], [11], [12], [14], [15], [23], [27])

1.3 오픈타입과 AAT: 더욱 향상된 폰트 처리 기술

오픈타입은 트루타입과 포스트스크립트를 아우르는 폰트의 형식이기도 하지만 더욱 세련된 폰트 처리 기술을 의미하기도 한다. AAT는 Mac OS X에서만 지원되는 폰트 처리 기술이다.

3. 포스트스크립트 타입 1 폰트를 트루타입 폰트로 변환하는 것은 수학적으로 가능한 일이지만, 트루타입 폰트는 기본적으로 2,048 단위의 그리드를 사용하고 타입 1 폰트는 1,000 단위의 그리드를 사용하고 있기 때문에 아웃라인의 둥글게 다듬는 부분에서 미묘한 차이가 발생한다고 한다.

그림 2. 하나의 폰트에서 글리프를 다양하게 대체한 예

오픈타입 오픈타입은 하나의 폰트에 이론적으로 65,536개[4]의 글리프를 포함할 수 있는데, 이 안에는 다양한 리거처와 글리프 대체, 상황에 맞는 위/아래 첨자, 커닝 세팅, 실제로 그려진 작은 대문자 등도 포함할 수 있다. 특히 매력적인 기능은 바로 '글리프 대체' 인데 이것은 말 그대로 하나 또는 그 이상의 캐릭터를 식자할 때 그에 상응하는 글리프를 다양한 글리프 가운데 선택할 수 있도록 한 것이다. 어떤 캐릭터를 식자할 때 같은 폰트 내에서 디자이너가 자신의 입맛에 맞는 글리프를 골라 쓸 수 있는 여지가 생긴 것이다. (그림 2 참조)

한편 여러 개의 캐릭터를 특정한 글리프로 표현할 수 있는데 이것을 리거처(ligature)라 한다. 익숙한 `f+i`, `f+l` 외에도 다양한 리거처를 구현할 수 있게 되었다.

오픈타입 폰트를 구성하기 위해서는 기본 테이블 외에도 트루타입과 관계된 테이블, 포스트스크립트과 관계된 테이블, 비트맵과 관계된 테이블, 향상된(advanced) 타이포그래피 테이블 등이 필요하다. 이 향상된 타이포그래피 테이블에 바로 GSUB (glyph substitution data)과 GPOS (glyph positioning data) 테이블이 들어있는데 이로부터 오픈타입 폰트의 세련된 글리프 운용의 묘를 발휘할 수 있게 된다. GSUB 테이블에는 문자체계와 언어체계에 따라 글리프를 대체할 수 있는 정보가 담겨 있다. 내부적으로 문자와 언어체계가 결정되고 그에 맞는 FeatureList와 LookupList를 구현하는 식으로 표현된다. 룩업(lookup)은 어떤 글리프를 하나에 대응할 것이냐, 여러 개에 대응할 것이냐, 여럿 중의 하나에 대응할것이냐, 문맥을 고려하여 대응할 것이냐 등을 결정하는 값이다.

글리프 대체나 리거처 등은 오픈타입 레이아웃 태그 중 특성 태그에 의해 구현되는데, 어떤 언어에서 특정 폰트에 담긴 글리프를 어떻게 운용할 것이냐를 담아둔 정보라고 할 수 있다. 하나의 특성 태그는 네 글자짜리 영어 알파벳으로 이름을 붙이는데, `liga`는 표준 리거처, `smcp`는 작은 대문자, `onum`은 옛날 방식의 숫자, `cswh`는 맥락에 따른 스와시(contextual swash) 등이 그 예이다.[5]

4. 한글과컴퓨터 사에서 배포하고 있는 함초롬바탕의 경우 64,145개의 글리프를, 함초롬돋움의 경우 64,450개의 글리프를 갖고 있다. Code2000 폰트의 경우 63,546개의 글리프를 갖고 있다.

5. 더 자세한 것은 마이크로소프트 사의 오픈타입 관련 사이트를 참조하라. `http://www.microsoft.com/typography/otspec/featuretags.htm`

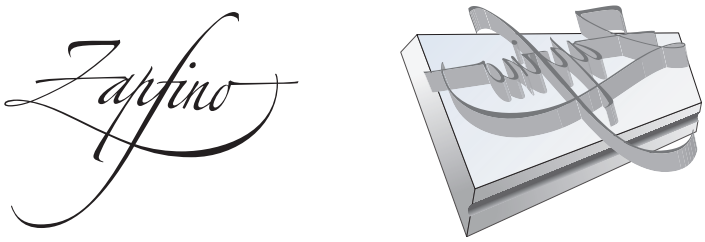


그림 3. Zapfino 폰트의 극단적인 리거처. 이를 활판 인쇄로 구현하려면?

AAT AAT는 애플사가 트루타입 폰트에서 발전시킨 트루타입 GX에서 한 걸음 더 나아간 폰트 기술이다. 멀티플 마스터의 기술과 트루타입 GX의 폰트 기술인 QuickDraw GX의 폰트 렌더링 방식을 개선하여 오픈타입 기술 못지 않은 글리프 대체, 작은 대문자, 상황에 맞는 위/아래 첨자, 숫자 모양, 다국어 처리, 다양한 리거처 등을 처리할 수 있다. AAT는 Mac OS X 운영체제에서만 구현이 가능하다.

이렇게 AAT와 오픈타입 폰트의 특징을 발판삼아 예전 활판 인쇄 시대에는 상상할 수 없었던 다양한 표현을 오픈타입을 지원하는 전문적인 전자출판 프로그램을 통해 이루어나가고 있다. 예를 들어 헤르만 자프(Herman Zapf)가 만든 Zapfino 폰트의 극단적인 리거처를 그림 3에서 볼 수 있다. 이것을 활판 인쇄 시절에 식자하려면 어떻게 해야했을까? 쾌걸 조로가 재빠르게 칼을 휘두르는 듯한 'Z' 때문에, 적분기호(∫)를 연상하게 하는 'f' 때문에 `Z+a+p+f+i+n+o`라는 독립된 일곱 활자를 결합하여 식자하는 것은 불가능하였을 것이다. 아마도 활판 인쇄 시절에는 Zapfino를 찍기 위하여 도장이나 관인과 같이 이 일곱 캐릭터를 조합하여 거꾸로 새긴 독립된 하나의 활자(글리프)를 만들어야 했을 것이다. 오픈타입은 이러한 Zapfino 리거처를 구현해줄 만큼 똑똑한 폰트이다.

1.4 XeTeX의 폰트 렌더링 방식

XeTeX은 폰트를 렌더링할 때 오픈타입과 AAT 폰트 기술을 모두 지원한다. 뒤에 나오겠지만 폰트 지정을 위해 `\fontspec` 명령을 사용할 때 `Renderer=AAT` 또는 `Renderer=ICU`를 옵션으로 줌으로써 각각의 폰트 기술을 사용할 수 있게 된다. AAT는 Mac OS X 운영체제에서만 구현 가능한 폰트 처리 기술이기 때문에 크로스 플랫폼을 겨냥한 오픈타입에 비해 상대적으로 제작된 폰트 수가 적다.

ICU (International Components for Unicode)는 유니코드 지원과 관련된 다양한 기능을 제공하는 라이브러리이다. ICU가 지원하는 것 중에 오픈타입 레이아웃이 들어있고 XeTeX은 이를 이용하여 오픈타입의 폰트 테이블 중 GSUB, GPOS 같은 것을 해석해내게 된다.

사실 XeTeX에서 옵션 `Renderer=AAT` (또는 `ICU`)는 명시적으로 지정해줄 필요가 없는데 이는 XeTeX이 알아서 AAT 방식의 폰트는 `Renderer=AAT`로, 오픈타입 방식의 폰트는 `Renderer=ICU`로 불러오기 때문이다.

한편 AAT와 ICU, 두 폰트 처리 기술을 모두 갖고 있는 폰트도 있다. Mac OS X에 기본으로 들어있는 일본어 폰트 히라기노(Hiragino)의 경우가 그 예이다. 앞서 말했지만 Mac OS X 운영체제는 AAT와 ICU 폰트 처리 기술을 모두 지원한다.

2 기본적인 사용법

이제 fontspec 패키지를 사용하여 어떻게 여러 폰트를 자유롭게 쓸 수 있는지 알아보자. 주요 내용은 fontspec 매뉴얼에서 가져왔다. 명령행이나 터미널에서 다음과 같이 입력하면 fontspec 매뉴얼을 볼 수 있다.

```
> texdoc fontspec
```

2.1 패키지 옵션

Fontspec 패키지를 사용하려면 당연히 `\usepackage[옵션]{fontspec}`과 같이 불러와야 한다. 패키지 옵션은 다음과 같다.

math fontspec 패키지의 기본 실행 옵션으로 수식을 조판할 때 텍의 기본 수식 폰트인 Computer Modern (CM)을 사용한다.

$$\lim_{n\to\infty}\left(\sin\theta-\cos\pi\int_0^x\frac{1}{x^2}\,dx+\sum_{k=1}^{n}\frac{1}{k(k+1)}\right)$$

위의 수식에서 lim, sin, cos 등 `\mathrm`으로 표현되는 사전예약 함수 이름은 현재 본문의 로만(roman, main) 폰트 (여기서는 Palatino)를 따라가지만 나머지 수식에 쓰인 숫자와 x, $\sum$, $\int$ 등 변수와 기호는 CM을 사용한다.

한편, mathpazo, mathptmx, concmath, mathdesign, unicode-math 등 별도의 수식 폰트 패키지를 얹으면 CM을 사용하지 않는다. 별도의 수식 폰트 패키지를 사용하고자 할 때는 fontspec보다 먼저 호출할 것을 권장하며, 특히 euler 수식 폰트 패키지는 반드시 fontspec보다 먼저 호출해야한다. 대부분의 수식 폰트 패키지는 fontspec과 같이 불러서 써도 큰 무리 없이 잘 처리될 것이다.

```
% mathptmx 수식 폰트 패키지를 불러온다.
\documentclass{article}
\usepackage{mathptmx}
\usepackage{fontspec}

\begin{document}
\[\lim_{n \rightarrow \infty}\left(\sin\theta
        -\cos\pi\int_0^x\frac{1}{x^2}\,dx
        +\sum_{k=1}^n \frac{1}{k(k+1)}\right)\]
\end{document}
```

$$\lim_{n\to\infty}\left(\sin\theta-\cos\pi\int_0^x\frac{1}{x^2}\,dx+\sum_{k=1}^{n}\frac{1}{k(k+1)}\right)$$

no-math \mathrm, \mathsf, \mathtt, \setboldmathrm에 대해서 본문 기본 폰트에 어울리게 조정하지 않고, 이마저도 CM의 것을 사용하겠다는 옵션이다. 간혹 수식 폰트가 어울리지 않게 바뀌는 것을 방지할 목적이다. 직관적으로 \setmainfot를 산세리프 계열의 Myriad Pro 폰트로 정하고 이 차이를 명확히 알아보자.

```
% math 옵션
\documentclass{article}
\usepackage[math]{fontspec} % 기본 실행 옵션
\setmainfont{Myriad Pro}

\begin{document}
\[\lim_{n \rightarrow \infty}\left(\sin\theta
        -\cos\pi\int_0^x\frac{1}{x^2}\,dx
        +\sum_{k=1}^n \frac{1}{k(k+1)}\right)\]
\end{document}
```

$$\lim_{n\rightarrow\infty}\left(\sin\theta-\cos\pi\int_0^x\frac{1}{x^2}\,dx+\sum_{k=1}^n\frac{1}{k(k+1)}\right)$$

```
% no-math 옵션
\documentclass{article}
\usepackage[no-math]{fontspec}
\setmainfont{Myriad Pro}

\begin{document}
\[\lim_{n \rightarrow \infty}\left(\sin\theta
        -\cos\pi\int_0^x\frac{1}{x^2}\,dx
        +\sum_{k=1}^n \frac{1}{k(k+1)}\right)\]\end{document}
```

$$\lim_{n\rightarrow\infty}\left(\sin\theta-\cos\pi\int_0^x\frac{1}{x^2}\,dx+\sum_{k=1}^n\frac{1}{k(k+1)}\right)$$

config 자주 쓰는 설정을 담아두는 사용자 정의 fontspec.cfg를 따로 만들 수 있다. 현재 작업하고 있는 폴더에 두거나 TDS (TEX Directory Structure) 의 적당한 위치에 두면 된다. 이 옵션은 fontspec 패키지를 불러오면 기본적으로 실행되는데 사용자의 TDS에 이미 아무 내용이 없는 fontspec.cfg가 들어있다. 명령행이나 터미널에서 다음과 같이 입력하면 어디에 들어있는지 알 수 있다.

```
> kpsewhich fontspec.cfg
```

no-config 사용자가 만들어둔 fontspec.cfg 환경 설정을 사용하지 않겠다는 옵션이다.

silent fontspec 패키지를 사용하다보면 폰트와 관련된 많은 경고를 만날 수 있다. 예를 들어 어떤 폰트 특성을 불러왔는데 해당 폰트에 그 특성이 없을 경우 경고가 생긴다. 이러한 경고를 .log 파일에는 기록하나 컴파일할 때는 보여주지 않는 옵션이다.

quiet 컴파일하면서 생긴 경고를 .log 파일에도 남기지 않는 옵션이다.

Fontspec 패키지와 같이 불러오면 좋은 패키지로 xunicode와 xltxtra가 있다. 자세한 것은 각각의 패키지 매뉴얼을 참고한다.

2.2 폰트를 불러오는 방법

본문에서 사용할 폰트를 불러오는 방법은 두 가지가 있다. 하나는 폰트의 이름을 부르는 것이고 다른 하나는 폰트의 파일 이름을 부르는 것이다.

2.2.1 폰트의 이름을 적는 법

폰트에는 고유의 이름이 있다. 글리프 모양과 어울리도록 이름을 짓기도 하고 폰트 디자이너의 이름을 모티브로 삼아서 짓기도 한다. 예를 들어 1932년 스탠리 모리슨(Stanely Morrison)은 영국의 일간지 《더 타임즈*The Times*》의 본문용 서체를 개발하면서 'Times New Roman'이라고 이름 붙였다.[6] 16세기 프랑스의 클로드 가라몽(Claude Garmond)은 세리프 활자를 제작하면서 자신의 이름을 따 'Garamond'라고 지었다.

다음은 어도비 사에서 만든 Minion Pro와 Myriad Pro 폰트 패밀리 가운데 가장 기본이 되는 모양과 굵기를 지닌 Regualr 폰트를 불러오는 예이다.

```
\fontspec{MinionPro-Regular}
        Adobe Minion Pro \newline
\fontspec{Minion Pro}
        Adobe Minion Pro \newline
\fontspec{MyriadPro-Regular}
        Adobe Myriad Pro \newline
\fontspec{Myriad Pro}
        Adobe Myriad Pro
```

Adobe Minion Pro
Adobe Minion Pro
Adobe Myriad Pro
Adobe Myriad Pro

한글 폰트의 경우 `\fontspec`대신 XeTeX-k*o*에서 정의한 `\hangulfontspec`을 쓴다.

```
\hangulfontspec{UnBatang}
        은 바탕 \newline
\hangulfontspec{은 바탕}
        은 바탕 \newline
\hangulfontspec{함초롬바탕}
        함초롬바탕 \newline
\hangulfontspec{08서울남산체 L}
        08서울남산체 L
```

은 바탕
은 바탕
함초롬바탕
08서울남산체 L

2.2.2 폰트 파일 이름을 적는 법

말 그대로 폰트의 확장자까지 포함하여 파일 이름을 직접 불러오는 방법이다. 한글 폰트의 경우 파일 이름이 한글로 되어 있으면 불러오지 못하니 주의해야 한다.

```
\hangulfontspec[%
        BoldFont=aritaB.ttf]{aritaL.ttf}
        아리따 L \newline
\textbf{아리따 B} \newline
\hangulfontspec{hgrgl.ttf}
        한겨레결체
```

아리따 L
아리따 B
한겨레결체

6. Times New Roman 이전에 《타임즈》를 찍어내던 활자를 Times Old Roman이라 한다. 애초에 활자로 제작된 Times New Roman을 여러 폰트 회사에서 디지털화하는 과정에서 모양과 이름이 약간씩 다르게 변했다.

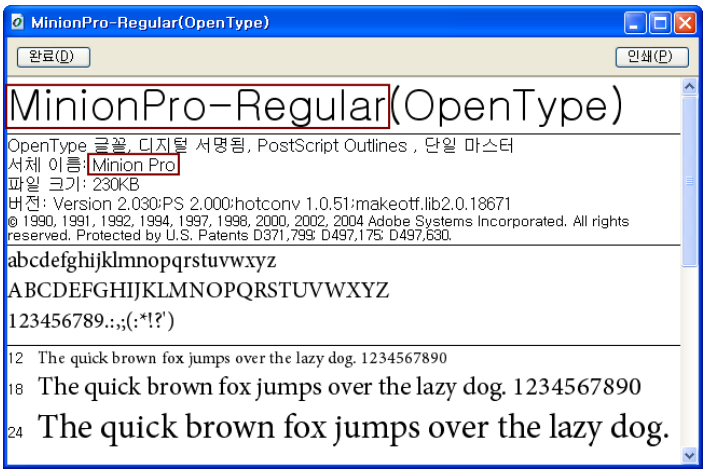


그림 4. 마이크로소프트 윈도에서 폰트 이름 알아보기

```
\fontspec[%
BoldFont = MinionPro-Semibold.otf,
ItalicFont = MinionPro-It.otf,
BoldItalicFont = MinionPro-SemiboldIt.otf]
        {MinionPro-Regular.otf}
Adobe Minion Pro \newline
\textbf{Adobe Minion Pro} \newline
\textit{Adobe Minion Pro}
```

Adobe Minion Pro
Adobe Minion Pro
Adobe Minion Pro

확장자를 미리 선언해놓고 폰트를 부르거나 특정 폴더에 있는 폰트를 불러올 수도 있다. Times New Roman의 보통/ 볼드/ 이탤릭/ 볼드이탤릭 폰트 패밀리가 각각 `times.ttf`, `timesbd.ttf`, `timesi.ttf`, `timesbi.ttf`이면 다음과 같이 지정하여 쓸 수 있다. 다음 소스 코드에서 별표(*)는 폰트 파일 이름에 공통적으로 들어가는 접두부 "times"를 축약한 것이다. 즉 `*bd`는 `timesbd`를 의미한다.

```
\fontspec[Extension = .ttf,
        BoldFont = *bd, ItalicFont = *i,
        BoldItalicFont = *bi]
        {times}
Times New Roman \newline
\textbf{Times New Roman} \newline
\textit{Times New Roman} \newline
\textbf{\textit{Times New Roman}}
```

Times New Roman
Times New Roman
Times New Roman
Times New Roman

2.2.3 폰트 관련 유틸리티

자신의 컴퓨터에 들어있는 폰트의 이름을 어떻게 알아낼 수 있을까? 운영체제별로 약간 차이가 있다. 여기서는 마이크로소프트 윈도 및 Mac OS X의 기본 폰트 뷰어와 몇 가지 폰트 유틸리티를 소개한다.

마이크로소프트 윈도 사용자의 경우 `.ttf`나 `.otf`를 더블 클릭하면 FONTVIEW 유틸리티가 실행되면서 그림 4와 같은 화면을 띄워준다. 거기서 크게 보이는 폰트 이름을 적거나

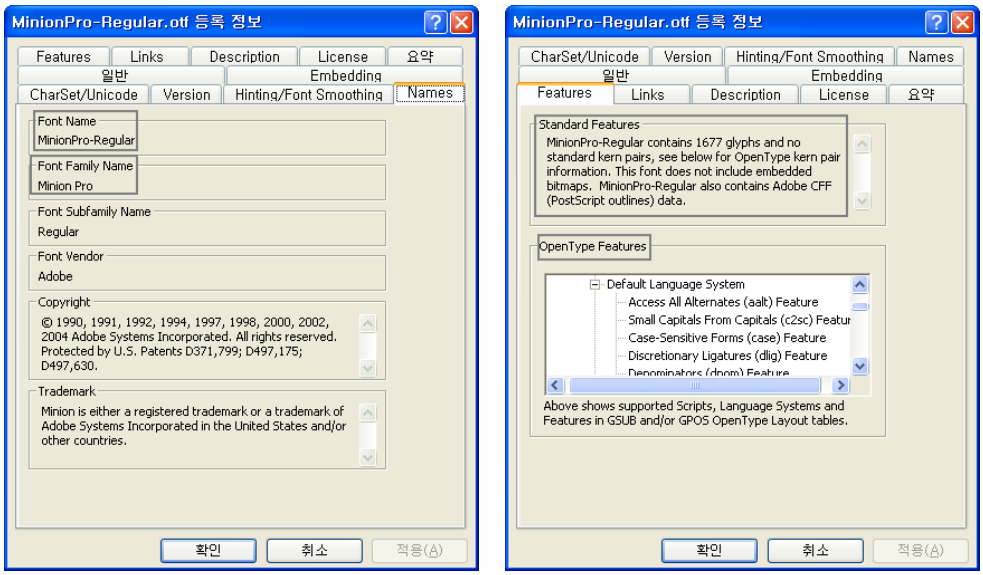


그림 5. FONT PROPERTIES EXTENSIONS에서 폰트 이름 알아보기

[서체 이름]에 나온 폰트 이름을 그대로 적어주면 된다. [서체 이름] 옆에 있는 것은 폰트 패밀리, 즉 이 폰트가 속한 글꼴 가족의 이름을 보여준다. 이 폰트 패밀리 이름을 적으면 폰트 패밀리 가운데 두께나 장평 등에서 가장 기본이 되는 폰트—일반적으로 Regular가 붙거나 폰트 패밀리 자신의 이름인 폰트—를 찍어준다. 앞의 예에서 `\fontspec{Minion Pro}`와 같이 Adobe Minion Pro 폰트 패밀리를 불렀을 때 가장 기본이 되는 MinionPro-Regular 폰트로 식자되었다. 띄어쓰기도 그대로 따라적어야 함을 유의한다.

마이크로소프트 사 홈페이지[7]에서 폰트 속성을 더 자세히 알려주는 프리웨어 유틸리티 FONT PROPERTIES EXTENSIONS를 다운로드할 수 있다. 이 유틸리티를 설치하고 `.ttf`나 `.otf`를 마우스 오른쪽으로 클릭하여 [속성]을 들여다보면 좀더 자세한 폰트 속성을 보여준다. [Names] 탭을 보면 해당 폰트의 이름과 해당 폰트가 속한 폰트 패밀리 이름도 보여준다. 그림 5는 `MinionPro-Regular.otf`의 속성을 보여주는 그림이다. FONTVIEW와 마찬가지로 이 폰트의 이름은 MinionPro-Regualr이고 폰트 패밀리 이름은 Minion Pro임을 보여주고 있다. 또 해당 폰트의 일반적인 정보와 오픈타입 특성 태그에 어떤 것이 내재되어 있는지 보여준다.

NEXUSFONT[8]는 시스템에 저장되어 있는 폰트의 설치 유무를 가리지 않고 자세한 폰트 정보를 알려주는 프리웨어 유틸리티이다. 폰트 이름을 복사할 수 있는 기능, 모든 글리프의 모양을 담은 문자표, 한글 폰트의 영문 이름 등을 알 수 있다. (그림 6 참조)

Mac OS X 사용자의 경우 기본 응용 프로그램에 들어있는 서체 관리자(FONT BOOK)라는 유틸리티를 사용하여 폰트 이름을 알아낼 수 있다. 어떤 폰트를 클릭하고 [미리보기]-[서체

7. `http://www.microsoft.com/typography/FreeToolsOverview.mspx`
8. `http://xiles.net/programs`

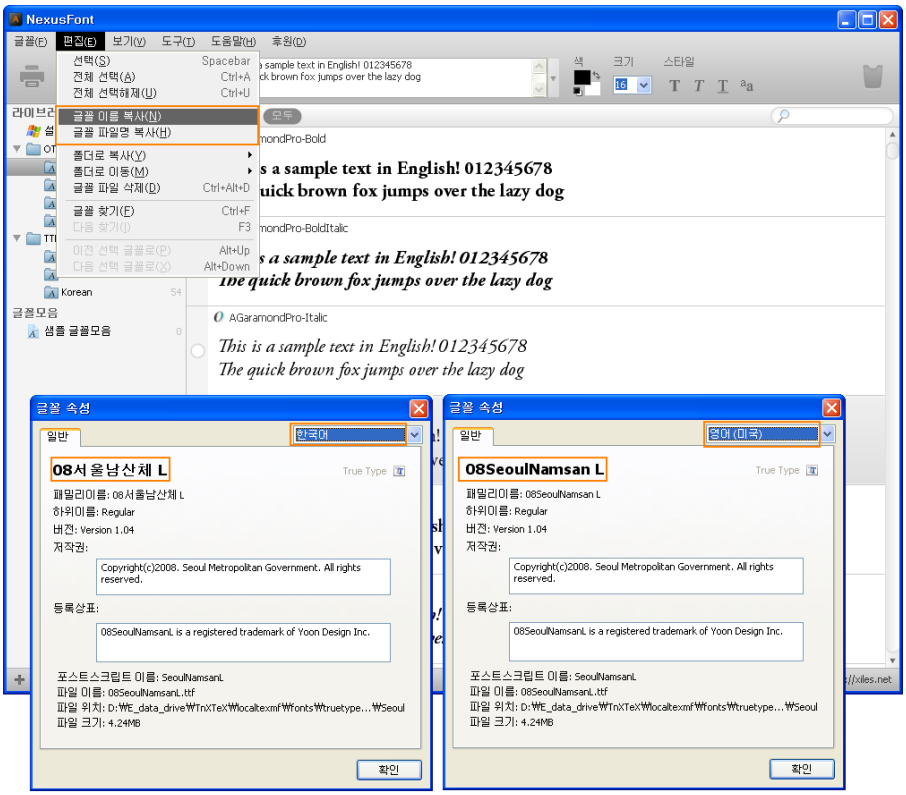


그림 6. NEXUSFONT에서 폰트 이름 알아보기

정보 보기]를 누르거나 command(⌘)+I를 누르면 그림 7과 같이 해당 폰트의 특성을 보여준다. 여기서 'PostScript name' 이나 'Full name' 에 나타난 폰트 이름을 적어주면 된다.

FONTMATRIX[9]는 Mac OS X는 물론 리눅스, 윈도 운영체제에서 사용할 수 있는 프리웨어 유틸리티이다. 앞서 소개한 다른 폰트 유틸리티와 비슷하게 글꼴 이름을 잘 알려주고, 시스템 설치 유무에 관계 없이 저장된 모든 폰트를 볼 수 있다. 또 모든 글리프의 내용을 pdf로 출력해주는 기능도 있다. (그림 8 참조)

2.3 본문에서 쓸 글꼴 지정하기

2.3.1 대표 글꼴 지정하기

본문에서 쓸 라틴 계열의 대표 폰트 세 가지—Roman (main), Sans Serif, Mono-spaced—를 다음과 같이 지정할 수 있다. \setmainfont는 예전 버전과의 호환을 위해 \setromanfont로 써도 관계없다.

```
\setmainfont[폰트에 부여할 옵션]{메인(로만) 폰트}
\setsansfont[폰트에 부여할 옵션]{산세리프 폰트}
\setmonofont[폰트에 부여할 옵션]{모노(고정폭) 폰트}
```

9. http://fontmatrix.net

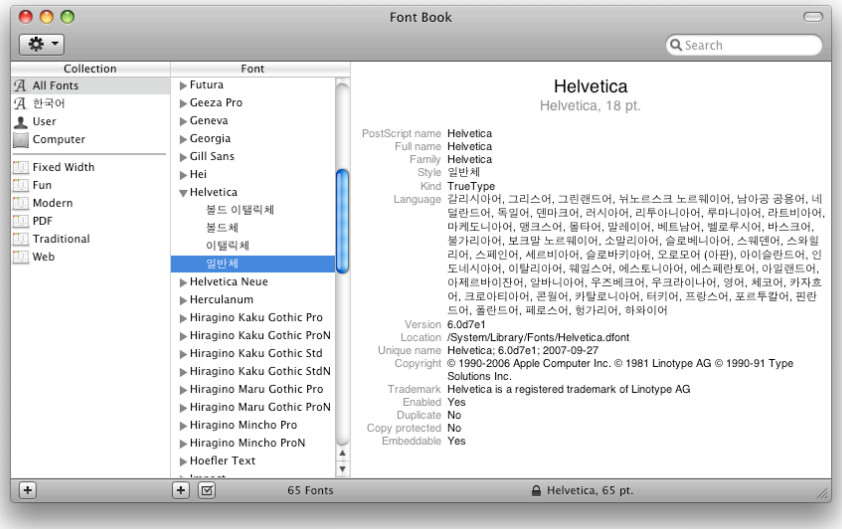


그림 7. 맥 OS X의 FONT BOOK에서 폰트 이름 알아보기

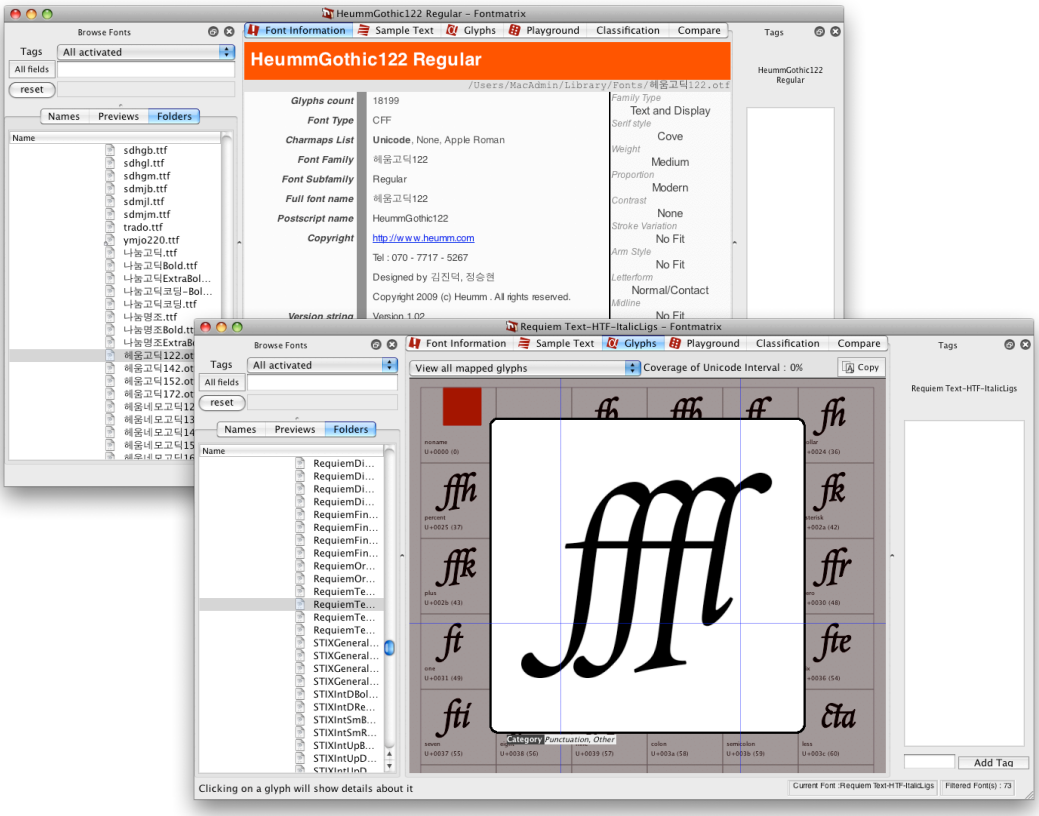


그림 8. FONTMATRIX에서 폰트 이름 알아보기

한글 문서를 조판할 때도 앞의 라틴 대표 폰트 지정방식과 유사하게 한글 대표 글꼴을 지정할 수 있다. 이 명령은 XeTeX-k*o*에서 정의된 것이다.

```
\setmainhangulfont[옵션]{한글 바탕 글꼴}
\setsanshangulfont[옵션]{한글 돋움 글꼴}
\setmonohangulfont[옵션]{한글 고정폭 글꼴}
```

한자 영역을 조판할 때도 XeTeX-k*o*에서 정한 대로 이 대표 폰트 방식을 따라간다.

```
\setmainhanjafont[옵션]{한자 바탕 글꼴}
\setsanshanjafont[옵션]{한자 돋움 글꼴}
\setmonohanjafont[옵션]{한자 고정폭 글꼴}
```

2.3.2 이따금 쓰는 글꼴 지정하기

대표 글꼴 외에도 본문에서 사용하고 싶은 폰트가 있을 것이다. 이럴 때는 `\newfontfamily` 또는 `\newfontface` 명령이 유용하다. `\newfontfamily`로 부른 폰트가 폰트 패밀리 이름일 경우 자동적으로 볼드, 이탤릭, 볼드이탤릭을 쓸 수 있다. `\newfontface`로 부른 폰트는 그 폰트 딱 하나만 쓸 수 있고 `\textbf`, `\emph` 등은 쓸 수 없다.

```
\newfontfamily\MyGentleFamily[%
        Mapping=tex-text]{Tekton Pro}
\MyGentleFamily Tekton Pro: \\
Regular \textbf{Bold} \textit{Italic}
\textbf{\textit{Bold Italic}}

\newfontface\MyFunny[%
        Mapping=tex-text]{Harrington}
\MyFunny Pros \& Cons of Hitchhikng
```

Tekton Pro:

Regular **Bold** *Italic* ***Bold Italic***

Pros & Cons of Hitchhikng

한글 글꼴의 경우 XeTeX-k*o*에 있는 `\newhangulfontfamily`와 `\newhangulfontface` 명령을 사용하면 된다.

```
\newhangulfontfamily\Hamchorom{함초롬바탕}
\Hamchorom 지금 눈 나리고 \\
\textbf{매화향기 홀로 아득하니} \\
\newhangulfontface\Bluebird{-파랑새L}
\Bluebird 내 여기 가난한 노래의 씨를 뿌려라
```

지금 눈 나리고
매화향기 홀로 아득하니
내 여기 가난한 노래의 씨를 뿌려라

2.3.3 글꼴 특성 지정하기

갖가지 폰트를 쓰면서 공통적으로 쓰는 폰트 특성은 `\defaultfontfeatures`에 담아둘 수 있다. 또 폰트를 지정할 때 미처 특성을 부여하지 못했더라도 `\addfontfeature(s)`를 이용하면 현재 폰트에 특성을 추가할 수 있다. 다음 소스에서 `\HL` 명령은 특정 부분을 색깔로 강조하기 위해 사용자가 지정한 명령어이다.

```
        \newcommand\HL[2][DarkRed]{\textcolor{#1}{#2}}
```

표 1. Size 옵션의 폰트 크기 범위

Size	s	크기 범위
x–	$s \geq x$	x 이상
x–y	$x \leq s < y$	x 이상 y 미만
–y	$s < y$	y 초과
z	$s = z$	딱 z 크기

```
\defaultfontfeatures{Mapping=tex-text}
\fontspec{Nueva Std}
Ausl\HL{\"a}nder\HL{---}Pâté \\
expos\HL[Teal]{é} \quad ros\HL{\'e} \\
\HL[Teal]{Ø}re \quad \HL{\O}re \quad
\HL[Teal]{Pâté} \quad \HL{P\^at\'e}
```

Ausländer—Pâté

exposé rosé

Øre Øre Pâté Pâté

```
\fontspec{Arno Pro}
Das ist Bleistift.\\
\addfontfeatures{Ligatures=Rare}
Das i\HL{st} Blei\HL{st}i\HL[Teal]{ft}.\\
\addfontfeatures{Letters=SmallCaps}
D\HL[Teal]{as ist} B\HL[Teal]{leistift}.
```

Das ist Bleistift.

Das ist Bleistift.

DAS IST BLEISTIFT.

2.3.4 글꼴 모양 지정하기

사용자가 잘 갖추어진 한 벌의 폰트 패밀리를 사용하면 볼드/ 이탤릭/ 볼드이탤릭 등에 대해 걱정할 필요가 없다. 그러나 폰트 패밀리를 찾기 어려워 각각의 모양을 서로 다른 폰트로 지정해야할 일이 있을 것이다. 그럴 때는 옵션으로 지정할 수 있다. 다음 폰트는 보통에 `Stockholm Std`, 이탤릭에 `Quill Std`, 볼드에 `AT Uncial Std`를 사용하여 이를 MyGirl이라는 폰트 패밀리로 묶은 것이다. 모양 지정 옵션은 `BoldFont`, `ItalicFont`, `BoldItalicFont`, `SlantedFont`, `BoldSlantedFont`, `SmallCapsFont`가 있다.

```
\newfontfamily\MyGirl[%
        BoldFont=AT Uncial Std,
        ItalicFont=Quill Std]
        {Stockholm Std}
\MyGirl Ground Control to \textit{Major} \textbf{Tom}
```

Ground Control to *Major* **Tom**

특히나 한글 글꼴은 라틴 폰트에 비해 글꼴 가족을 찾기가 힘들다. 이럴 때 임의의 글꼴 가족을 꾸려 유용하게 사용할 수 있을 것이다. 모양 지정 옵션은 `\set{main/sans/mono}font`, `\{hangul}fontspec`, `\new{hangul}fontfamily`에도 그대로 적용할 수 있다.

2.3.5 크기에 따라 글꼴 및 특성을 지정하기

`SizeFeatures` 특성의 `Size` 옵션을 사용하면 폰트의 크기에 따라 폰트를 따로 부를 수 있고 특성도 따로 지정할 수 있다. 폰트 크기를 s라 하면 표 1과 같이 세분화할 수 있다.

```
\fontspec[SizeFeatures={%
  {Size={-9}, Font={Helvetica Neue}, Color=DarkRed},
  {Size={9-12},Color=DimGray},
  {Size={12-14}, Font={Trajan Pro}, Color=Olive},
  {Size={14-}, Font={Nueva Std}, Color=Navy}}]{Gentium}
{\footnotesize To die,} \\
{\normalsize to sleep---} \\
{\large To sleep,} \\
{\LARGE perchance to dream.} \par
\raggedleft
  ---The soliloquy of \emph{Hamlet}
```

To die,
to sleep—
TO SLEEP,
perchance to dream.
—The soliloquy of Hamlet

3 폰트에 구애받지 않는 특성

3.1 컬러

`Color` 옵션은 앞에서 예를 들었다. 원래 fontspec 이전 버전에서는 0066FF처럼 00부터 FF까지 16진수 두 자리로 이루어진 세 쌍의 RGB 값으로 지정된 컬러만을 지원했었다. 최신 fontspec은 xcolor 패키지의 옵션으로 `dvipsnames`, `svgnames`, `x11names`를 쓰면 사전에 정의된 많은 컬러 이름을 바로 불러다 쓸 수 있다. color 패키지는 지원하지 않는다. 한편 00부터 FF까지 RGB 값을 한 쌍 더 붙이면 투명도(transparency)를 지정할 수 있다. 00은 완전히 투명한 것이고 FF는 색깔이 꽉 메워진 것이라 생각하면 된다.

```
\fontspec{Cronos Pro Bold}
{\addfontfeature{Color=3333CC88}K}
        \kern-.9ex\lower.15em\hbox{%
{\addfontfeature{Color=EE000088}T}}
        \kern-.8ex\lower-.15em\hbox{%
{\addfontfeature{Color=99CC3388}U}}
        \kern-.85ex
{\addfontfeature{Color=1E90FF88}G}
```

그런데 위의 코드를 직접 컴파일해 보면 투명도가 구현되지 않을 것이다. 그 이유는 XeTeX의 기본 드라이버인 xdvipdfmx가 아직 투명도를 지원하지 않기 때문이다. 투명도를 구현하려면 Mac OS X 운영체제에서만 실행되는 또 다른 드라이버 xdv2pdf를 이용해야만 하는데, 컴파일 방법은 다음과 같다.

```
> xelatex -output-driver="xdv2pdf" filename.tex
```

투명도는 `Opacity`라는 옵션을 이용해 0(투명)에서 1(불투명) 사이의 값을 줄 수도 있다.

```
\fontspec{Cronos Pro Bold}
{\addfontfeature{Color=Navy, Opacity=.5}K}
        \kern-.9ex\lower.15em\hbox{%
{\addfontfeature{Color=Crimson, Opacity=.5}T}}
        \kern-.8ex\lower-.15em\hbox{%
{\addfontfeature{Color=Olive, Opacity=.6}U}}
        \kern-.85ex
{\addfontfeature{Color=SkyBlue, Opacity=.6}G}
```

3.2 스케일

서로 다른 폰트를 지정해서 쓰다보면 두 폰트 디자인이 달라 같은 크기의 폰트에서도 어느 한쪽이 커보이거나 작아보일 수 있다. 이럴 때 Scale 옵션으로 어느 한쪽을 다른 폰트와 크기가 비슷하게 조정할 수 있다. Scale 옵션은 숫자로 조정가능한데 예를 들어 0.5는 현재 상태의 50퍼센트로 축소하고 1.2는 현재보다 120퍼센트 확대한다는 의미이다. 또한 Scale 값으로 MatchLowercase와 MatchUppercase를 지정할 수 있다. 이는 스케일을 현재 로만 폰트로 지정된 폰트의 소문자(lowercase)와 대문자(uppercase)에 맞춘다는 의미이다. 숫자로 스케일을 조정하는 것보다 더 편리한 기능이다.

```
\newcommand\sampletexttwo{It is a \textsf{roaring} silence.}

\drawfontframe{%
\setsansfont[Color=DarkRed]
  {Helvetica Neue}\sampletexttwo}
\drawfontframe{%
\setsansfont[Scale=MatchUppercase,Color=DarkRed]
  {Helvetica Neue}\sampletexttwo}
\drawfontframe{%
\setsansfont[Scale=MatchLowercase,Color=DarkRed]
  {Helvetica Neue}\sampletexttwo}
\drawfontframe{%
\setsansfont[Scale=.7,Color=DarkRed]
  {Helvetica Neue}\sampletexttwo}
```

It is a roaring silence.

It is a roaring silence.

It is a roaring silence.

It is a roaring silence.

앞의 소스 코드에서 사용한 \drawfontframe는 layouts 패키지에 들어있는 명령으로 베이스라인과 참조점(reference point)을 문자열에 찍어준다.

```
\newcommand\sampletextone{이것은 \textsf{소리 없는} 아우성}

\setsanshangulfont[Scale=MatchUppercase,
  Color=Teal]{나눔고딕}\sampletextone

\setsanshangulfont[Scale=MatchLowercase,
  Color=Teal]{나눔고딕}\sampletextone

\setsanshangulfont[Scale=.7,
  Color=Teal]{나눔고딕}\sampletextone
```

이것은 소리 없는 아우성

이것은 소리 없는 아우성

이것은 소리 없는 아우성

3.3 단어 간격과 자간

어떤 폰트를 불러놓고 보니 단어 간격(interword space)이 너무 성기거나 촘촘하여 마음에 들지 않을 수 있다. 이때 WordSpace 옵션을 이용하여 단어 간격을 조절할 수 있다. 모두 세 개의 인수를 받을 수 있는데 WordSpace={x,y,z}에서 x는 기본 단어 간격, y는 기본 단어 간격에서 늘어날 수 있는 허용치, z는 기본 단어 간격에서 줄어들 수 있는 허용치를 의미한다. WordSpace={x}는 WordSpace={x,x,x}와 같은 의미이다. 물론 단어 간격과 관련하여 TEX의 원시명령 \spaceskip도 있음을 상기하자.

```
\fontspec[WordSpace=3.0]{Adobe Garamond Pro}
Allman Brothers Band \\
\addfontfeatures{WordSpace={1.0,1.5,2.0}}
Allman Brothers Band \\
\addfontfeatures{WordSpace={.5,.3,.7}}
Allman Brothers Band
```

Allman Brothers Band
Allman Brothers Band
Allman Brothers Band

라틴 폰트의 경우 자간을 인위적으로 조절하는 것은 삼가야 한다. 왜냐하면 이미 폰트에 적절한 커닝(kerning)이나 트래킹(tracking)을 집어넣었기 때문이다. 폰트 디자이너의 디자인 의도를 존중해야한다. 그럼에도 불구하고 자간을 조절할 필요가 있을 때에는 LetterSpace 옵션을 사용한다. LetterSpace=x로 설정하면 x 퍼센트 ($\frac{\text{현재 폰트 크기}\times x}{100}$) 만큼 본문 크기에 비례하여 간격을 띄운다. 예를 들어 본문 10포인트 크기에서 LetterSpace=1.0으로 설정하면 $\frac{10\text{pt}\times 1.0}{100} = 0.1\text{pt}$만큼 자간을 띄운다.

```
\fontspec{Gentium}
\addfontfeature{LetterSpace=0.0}
STAIRWAY TO HEAVEN \\
\addfontfeature{LetterSpace=5.0}
STAIRWAY TO HEAVEN
```

STAIRWAY TO HEAVEN
STAIRWAY TO HEAVEN

한글 글꼴의 경우 사진식자 시대 이후 소위 '마이너스 자간'을 적용하는 것이 인쇄출판업계의 오랜 관행이었다. 자간을 줄이지 않고 사용할 경우 글자와 글자가 듬성듬성하여 판면이 너무 성긴 느낌을 받았기 때문이다. XeTeX-k*o*에서는 interhchar라는 옵션을 주어 자간을 조절할 수 있다. 단, 모노(고정폭) 글꼴 지정 명령에서는 적용되지 않는다. 한글 글꼴의 모양에 따라 최적화된 자간값이 다를 수 있으므로 주의해서 사용해야한다. 일반적으로 한글 글꼴의 타이포그래피는 XeTeX-k*o*의 기본값을 존중하면 될 것이다.

```
\hangulfontspec[interhchar=-.1em]{나눔명조}
며느리밥풀꽃에 대한 보고서 \\
\hangulfontspec[interhchar=0em]{나눔명조}
며느리밥풀꽃에 대한 보고서 \\
\hangulfontspec[interhchar=.2em]{나눔명조}
며느리밥풀꽃에 대한 보고서
```

며느리밥풀꽃에 대한 보고서
며느리밥풀꽃에 대한 보고서
며 느 리 밥 풀 꽃 에 대 한 보 고 서

3.4 장평, 기울이기, 굵게 하기

하나의 폰트에는 폰트 디자이너가 의도한 폰트 디자인이 담겨 있다. 이를 사용자가 인위적으로 변경하는 것은 삼가야 한다. 그러나 사진식자 이후로 한글 글꼴은 앞서 언급한 마이너스 자간과 함께 장체를 쓰는 것이 일반적이었다. 장체(長體)란 글자의 가로길이를 세로길이보다 줄여 홀쭉하게 만드는 것을 의미한다. 반대로 평체(平體)란 글자의 세로길이를 가로길이보다 줄여 통통하게 만드는 것을 의미한다.

제대로 된 한벌의 글꼴 가족이라면 본문 바탕 글꼴을 기준으로 굵은 것과 가는 것, 좁은 것과 넓은 것을 가지고 있을 것이다. 그러나 많은 이유가 있겠지만, 한글 글꼴은 라틴 폰트보다 글꼴 가족을 제대로 갖추지 못한 경우가 많다. 이럴 경우 인위적으로 조절된 장과 평, 기울이거나 굵은 글꼴을 만들 수 있다. 단 글꼴의 품위는 담보하기 어렵다.

표 2. 어도비 사에서 제안한 용도별 시각적 폰트 크기

용도	크기	용도	크기
캡션(caption)	6–8포인트	본문(regular)	9–13포인트
작은 제목(subhead)	14–24포인트	큰 제목(display)	25–72포인트
포스터(poster)	72포인트 초과		

```
\hangulfontspec{제주명조OTF}
오늘도 조용히 일어나 \\
\hangulfontspec[FakeSlant=0.2]{제주명조OTF}
혼자 걷는 너에게 \\
\hangulfontspec[FakeStretch=.8]{제주명조OTF}
나는 이렇게 부르지 \\
\hangulfontspec[FakeStretch=1.2]{제주명조OTF}
저 파란 하늘 위에 \\
\hangulfontspec[FakeBold=1.5]{제주명조OTF}
날으는 법을 배우는 작은새
```

오늘도 조용히 일어나
혼자 걷는 너에게
나는 이렇게 부르지
저 파란 하늘 위에
날으는 법을 배우는 작은새

한편 AutoFakeBold와 AutoFakeSlant라는 옵션이 있다. 이것을 사용하면 fontspec이 알아서 볼드/ 이탤릭/ 볼드이탤릭 폰트를 만들어준다.

```
\fontspec[AutoFakeSlant=.2,
        AutoFakeBold=1.8]{Stockholm Std}
Used to say I like Chopin\\
\textbf{Love me now and again} \\
\textit{Rainy days never say goodbye} \\
\textbf{\textit{To desire
        when we are together}}
```

Used to say I like Chopin
Love me now and again
Rainy days never say goodbye
To desire when we are together

3.5 시각적 크기

잘 만들어진 폰트 패밀리는 폰트 크기에 따라 불러오는 폰트가 다르다. 같은 폰트에 대해서 캡션용, 본문용, 작은 제목용, 큰 제목용으로 나누어 놓고 사용자가 폰트 크기를 조절하면 그에 알맞은 폰트를 가져다 써준다. 쉽게 말하면 본문용(9~13포인트) 폰트를 약 25포인트 크기의 큰 제목에 사용하면 너무 진한 느낌이 들고 7포인트의 캡션에 사용하면 너무 흐린 느낌이 드는 것을 방지하고자 하는 것이다. 시각적 크기에 대해 어도비 사에서 제시한 가이드는 표 2와 같다.

XeTeX은 시각적 크기별로 구성된 폰트 패밀리를 불러올 경우 자동으로 해당 크기에 어울리는 폰트를 지정해준다. 다음 Brioso Pro 폰트 패밀리의 예를 보자. 폰트 패밀리를 지정하고 크기를 달리하여 텍스트를 입력하면 실제로 \footnotesize에 대해 BriosoPro-Capt.otf, \normalsize에 대해 BriosoPro-Regular.otf, \LARGE에 대해 BriosoPro-Subh.otf, \Huge에 대해 BriosoPro-Disp.otf를 자동으로 지정한다.

```
\fontspec{Brioso Pro}

{\footnotesize Ziggy Stardust} \\
{\normalsize Ziggy Stardust} \\
{\LARGE Ziggy Stardust} \\
{\Huge Ziggy Stardust}
```

Ziggy Stardust
Ziggy Stardust
Ziggy Stardust
Ziggy Stardust

실제 폰트의 차이를 보자. 시각적 크기에 맞춰 따로 제작된 폰트를 비교하기 위하여 모두 같은 크기에 놓고 텍스트를 조판해보았다. 시각적 크기가 적용되지 않도록 하려면 각각의 폰트 이름을 불러올 때 `OpticlaSize=0` 옵션을 주거나 '폰트 파일 이름'으로 불러오면 된다. 본문 폰트를 기준으로 놓고 볼 때 캡션 폰트는 너무 흐리지 않도록 글리프가 조금 진하고 옆으로 퍼져있다. 큰 제목 폰트는 너무 진하고 퍼져있는 느낌을 주지 않도록 조금 홀쭉하고 촘촘하게 디자인 되어 있다.

```
\fontspec[OpticalSize=0]{BriosoPro-Capt}
        Canzona (There will be time) \\
\fontspec[OpticalSize=0]{BriosoPro-Regular}
        Canzona (There will be time) \\
\fontspec[OpticalSize=0]{BriosoPro-Subh}
        Canzona (There will be time) \\
\fontspec[OpticalSize=0]{BriosoPro-Disp}
        Canzona (There will be time)
```

Canzona (There will be time)
Canzona (There will be time)
Canzona (There will be time)
Canzona (There will be time)

```
\fontspec{CronosPro-Capt.otf}
        I shot the Sheriff \\
\fontspec{CronosPro-Regular.otf}
        I shot the Sheriff \\
\fontspec{CronosPro-Subh.otf}
        I shot the Sheriff \\
\fontspec{CronosPro-Disp.otf}
        I shot the Sheriff
```

I shot the Sheriff
I shot the Sheriff
I shot the Sheriff
I shot the Sheriff

4 폰트에 좌우되는 특성

이제 ICU 오픈타입과 AAT로 나누어 폰트에 따라 좌우되는 특성을 알아보자. Fontspec 매뉴얼 가운데 주요한 것을 발췌하여 소개한다.

4.1 ICU 오픈타입 특성

4.1.1 숫자 모양

숫자 모양은 `Numbers` 특성으로 지정한다. 숫자의 높낮이에 따라 올드 스타일(`OldStyle/Lowercase`)과 라이닝(`Lining/ Uppercase`), 글자 폭에 따라 고정폭(`Proportional`)과 비율폭(`Monospaced`)으로 나뉜다. 영문자 O 또는 o와 헷갈리지 않도록 사선을 그은 숫자 `0`(`SlashedZero`)도 있다. 고정폭 숫자는 표에서 사용하면 위/아래의 자릿수가 일치하게 되어 가독성을 담보할 수 있다.

```
\newfontface\OldPropFigure[Numbers={OldStyle,Proportional}]{Arno Pro}
\newfontface\LiningProFigure[Numbers={Lining,Proportional}]{Arno Pro}
\newfontface\OldMonoFigure[Numbers={OldStyle,Monospaced}]{Arno Pro}
\newfontface\LiningMonoFigure[Numbers={Lining,Monospaced}]{Arno Pro}
\newfontface\SlashedZeroFigure[Numbers={SlashedZero}]{Arno Pro}

\begin{tabularx}{.95\textwidth}
        {>{\hsize=.4\hsize}X|>{\hsize=.3\hsize}X>{\hsize=.3\hsize}X}
\toprule & 올드 스타일 & 라이닝 \tabularnewline
\midrule
비율폭 & \Large\OldPropFigure 0123456789
      & \Large\LiningProFigure 0123456789 \tabularnewline
고정폭 & \Large\OldMonoFigure 0123456789
      & \Large\LiningMonoFigure 0123456789 \tabularnewline
사선을 그은 숫자 0 & --- & \Large\SlashedZeroFigure 0 \tabularnewline
\bottomrule
\end{tabularx}
```

	올드 스타일	라이닝
비율폭	0123456789	0123456789
고정폭	0123456789	0123456789
사선을 그은 숫자 0	—	Ø

```
\XeTeXuseglyphmetrics=0
\OldPropFigure %올드 스타일 비율폭
\multido{\i=0+1}{10}{\fbox{\i}}

\OldMonoFigure %올드 스타일 고정폭
\multido{\i=0+1}{10}{\fbox{\i}}
```

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

`OldStyle`과 `Linging`, `Proportional`과 `Monospaced`는 상충되는 개념이다. 폰트 특성을 지정할 때 `Numbers={OldStyle, Lining}`와 같이 상충되는 것을 지정하면 뒤에 적은 옵션(여기서는 `Lining`)만 적용된다. 그러면 올드 스타일이나 라이닝은 언제 사용할까? 다음 예문을 보자. 본문에서 연이어 숫자가 나올 때 라이닝 숫자가 조금 거추장스럽게 느껴질 수 있다. 이때 올드 스타일 숫자를 쓰면 본문과 조금 더 어울리는 느낌을 받는다.

```
\newcommand\sampletextthree{%
  Let me know your home address till 4 o'clock to send the gift.\\
  Me? My home address is, zip code 235-819, KTUG Street 746.}

\fontspec[Numbers=Lining]{Minion Pro} \sampletextthree \\
\fontspec[Numbers=OldStyle]{Minion Pro} \sampletextthree
```

Let me know your home address till 4 o'clock to send the gift.
Me? My home address is, zip code 235-819, KTUG Street 746.
Let me know your home address till 4 o'clock to send the gift.
Me? My home address is, zip code 235-819, KTUG Street 746.

4.1.2 리거처

리거처(합자)는 Ligatures 특성으로 정의할 수 있는데 보통(Common), 희귀하거나 임의로 지정된(Rare/ Discretionary), 맥락에 따른(Contextual) 리거처 등이 있다. 이 외에도 전통적인 텍 방식(TeX),[10] 역사적(Historic), 필수(Required) 리거처가 있다.[11] Common, Contextual, Required 리거처는 기본적으로 지정되는데 이를 없애기 위해서는 앞에 No를 붙여 NoCommon, NoContextual, NoRequired로 지정하면 된다.

```
\fontspec[Ligatures=NoCommon]{Arno Pro}
office flower fjord often \\
\fontspec[Ligatures=Common]{Arno Pro}
o\HL{ffi}ce \HL{fl}ower \HL{fj}ord o\HL{ft}en \\
\addfontfeatures{Ligatures=Rare}
introdu\HL{ct}ion  \HL{st}udent
```

office flower fjord often
office flower fjord often
introduction student

```
\fontspec{Junicode}
I II III IV <1> [2] [[3]] \\
\addfontfeatures{Ligatures=Rare}
I II III IV <1> [2] [[3]] \\
\addfontfeatures{Ligatures=Historic}
A\HL{ttr}ibution \HL{an}d \HL{st}ru\HL{gg}le
```

I II III IV <1> [2] [[3]]
I II III IV ❶ ② ③
Attribution and struggle

```
\fontspec[Ligatures=NoCommon]{Linux Libertine O}
Quest'uomo wurtzite \\
\fontspec[Ligatures={Common,Rare}]
        {Linux Libertine O}
\HL{Qu}est'uomo wur\HL{tz}ite \\
\addfontfeatures{Ligatures=Historic}
Di\HL[Teal]{st}ri\HL[Teal]{ct} No.\,9 \\
Liberta' per Que\HL{st}'Uomo
```

Quest'uomo wurtzite
Quest'uomo wurtzite
District No. 9
Liberta' per Quest'Uomo

다음은 XeTeX-k*o* 매뉴얼에 있는 유명한 예제이다.

```
\hangulfontspec{Adobe 명조 Std} 주식회사 \\
\hangulfontspec[Ligatures=Discretionary]
        {Adobe 명조 Std}
\disablehangulspacingandlinebreak 주식회사
```

주식회사
주식
회사

로버트 브링허스트(Robert Bringhurst)는 [16]에서 매우 희귀한 리거처 f+f+f+l가 쓰인 예로 독일어 단어 Sauerstoffflasche (oxygen bottle; 산소병)와 Sauerstoffflaschenspüller (oxygen bottle washer; 산소병 세척기)를 소개하였다. (그림 9 참조)

4.1.3 작은 대문자

작은 대문자는 이탤릭이나 볼드와 마찬가지로 특정 부분을 강조하기 위하여 사용한다. 작은 대문자는 대문자와 모양은 비슷하지만 소문자의 높이와 거의 동일하며 일반적으로 대문자의

10. Ligatures=TeX 옵션을 준 것은 Mapping=tex-text 옵션을 준 것과 같다.
11. Required 리거처는 아랍어와 시리아어를 조판할 때 쓰인다.

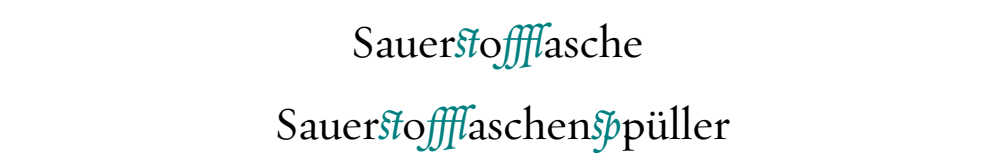


그림 9. Hoefler & Frere-Jones의 Requiem 폰트 이탤릭과 리거처

60내지 70퍼센트의 크기로 디자인한다. 대소문자의 구별이 없는 한글에는 작은 대문자가 없다. 잘 만들어진 폰트 패밀리에는 보통 작은 대문자가 포함되어 있다.

```
\fontspec{Warnock Pro}
The Phantom of the Opera is there. \\
\fontspec[Letters=SmallCaps]{Warnock Pro}
The Phantom of the Opera is there. \\
\fontspec[Letters=UppercaseSmallCaps]
        {Warnock Pro}
The Phantom of the Opera is there.
```

The Phantom of the Opera is there.
The Phantom of the Opera is there.
The Phantom of the Opera is there.

작은 대문자 효과를 주기 위하여 대문자를 인위적으로 줄여 사용할 경우 소문자와 잘 어울리지 않고 대문자를 단지 축소하였기 때문에 폭이 좁거나 굵기가 더 가늘어지기도 한다. 따라서 실제로 그려진 작은 대문자를 사용하여야 한다.

```
\fontspec[Letters=SmallCaps]{Brioso Pro}
        Old and Wise (true) \\
\fontspec{Brioso Pro}
        O\scalebox{.7}{LD AND}
        W\scalebox{.7}{ISE} (faux)
```

Old and Wise (true)
Old and Wise (faux)

작은 대문자는 문단의 첫 줄에 사용하여 독자의 시선을 끌기도 하고 본문 중간에 등장하는 약어에 사용하기도 한다. 올드 스타일 숫자와 같이 사용하면 더욱 세련된 조판 느낌을 줄 수 있다. 다음 예문을 비교해보자.

```
\fontspec{Minion Pro}
Leroy Jethro Gibbs is a fictional character from the NCIS television series
by CBS Television. Watch the TV tonight P.M. 10:00.

\addfontfeature{Numbers=OldStyle}
\textsc{Leroy Jethro Gibbs} is a fictional character from the \textsc{ncis}
television series by \textsc{cbs} Television.
Watch the \textsc{tv} tonight \textsc{p.m.} 10:00.
```

Leroy Jethro Gibbs is a fictional character from the NCIS television series by CBS Television. Watch the TV tonight P.M. 10:00.

Leroy Jethro Gibbs is a fictional character from the ncis television series by cbs Television. Watch the tv tonight p.m. 10:00.

4.1.4 커닝

커닝이란 인접한 두 글리프의 간격을 조정하여 시각적으로 보기 좋게 만드는 것이다. 잘 만들어진 폰트의 경우 커닝 쌍에 대한 테이블을 갖고 있고, 이 테이블에 등록된 글리프 쌍이 왔을

때 간격을 조정하게 된다. Fontspec 패키지는 폰트에 담긴 커닝 정보를 사용할 수 있도록 지원한다. `Kerning=Uppcase`는 대문자로 조판된 문자열이 오히려 답답하게 보일 수 있어 이를 보정하는 것이다. 다음 예에서 A와 V, 그리고 T와 y의 간격을 살펴보자.

```
\fontspec[Kerning=On]{Minion Pro}
Aviation Typography
\addfontfeature{Kerning=Off}
Aviation Typography

\fontspec[Letters=SmallCaps]{Warnock Pro}
Aviation Typography
\addfontfeature{Kerning=Off}
Aviation Typography

\fontspec{Myriad Pro}
AVIATION TYPOGRAPHY
\addfontfeature{Kerning=Uppercase}
AVIATION TYPOGRAPHY
```

Aviation Typography

Aviation Typography

Aviation Typography

Aviation Typography

AVIATION TYPOGRAPHY

AVIATION TYPOGRAPHY

4.1.5 특수 효과

앞에서 오픈타입 폰트의 특징을 설명하면서 잠깐 언급했지만 fontspec은 오픈타입 특성 태그를 직관적으로 이해하기 쉽도록 `\fontspec` 명령의 옵션으로 바꾸어놓은 것이다. `\fontspec` 명령을 사용하면서 `RawFeature`라는 옵션을 사용하면 특성 태그를 바로 불러올 수 있다. 예를 들어 표준 리거처의 특성 태그는 `liga`인데 `RawFeature=+liga`와 같이 옵션을 주면 `Ligatures=Common`으로 준 것과 동일하다.

오픈타입 특성 태그의 대부분은 오픈타입의 글리프 대체에 있다. 즉 하나의 캐릭터에 대해 하나 또는 여러 개의 글리프를 대응시켜놓고 사용자가 원하는 것을 찾아 꺼내쓸 수 있도록 한 것이 핵심이다.

사실 fontspec 매뉴얼에는 오픈타입의 특성을 앞서 설명한 `Ligatures`나 `Numbers` 외에도 `Contextuals`, `Stylistic Set`, `Character Variants`, `Alternates`, `Annotation`, `Style` 등으로 구분하여 자세히 설명하고 있다. 서구의 조판 관행에서 이러한 오픈타입 특성 태그의 구분은 충분한 의미가 있다. 폰트 기술의 집약체인 오픈타입 폰트—소위 전문가 타입—와 XeTeX 및 fontspec 패키지를 이용하여, 조판 상황에 어울리는 다양한 타이포그래피를 구현할 수 있는 상태에 이르게 된 것은 상당히 고무적인 일이라 아니할 수 없다.

그러나 서구 조판의 전통과 관행에 그다지 익숙하지 않은 글쓴이에게 이러한 오픈타입 특성 태그의 구분은 크게 가슴에 와닿지 않은데다 그 차이를 알아내려고 애쓰는 것이 이만저만한 고충이 아닐 수 없었다. 또 매뉴얼에 소개된 폰트 외에 같은 효과를 포함하고 있는 오픈타입 폰트를 찾는 것도 만만한 일은 아니었다. 따라서 이러한 특성 태그를 한데 모아 '특수 효과'라고 이름 붙이고 예제 위주로 설명하고자 한다. 대부분 '글리프 대체'에 관한 예제이다. (Fontspec 매뉴얼 제10장 참고)

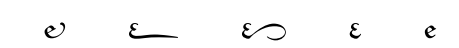
```
\fontspec{Warnock Pro} \itshape
Premiata Forneria Marconi

\addfontfeature{Contextuals=Swash}
Premiata Forneria Marconi; \\
P.F.M is a famous band.
```

Premiata Forneria Marconi
Premiata Forneria Marconi;
P.F.M is a famous band.

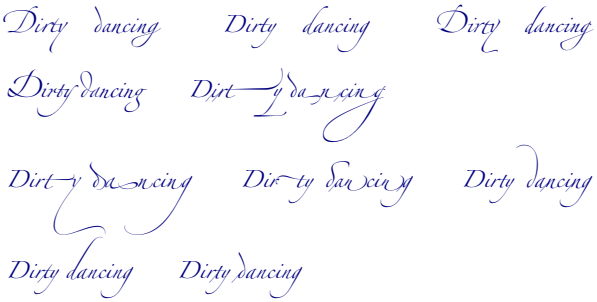
스와시는 우리말로 적절히 번역하기가 어렵다. 스와시는 이탤릭체에서 주로 사용되는데 단어의 맨 앞글자나 뒷글자를 장식하는 효과이다. 단어 중간에 스와시가 오면 앞뒤 철자와 간격이 벌어질 우려가 있어 오히려 보기 싫은 경우도 있다.

```
\multido{\i=0+1}{5}{%
    \fontspec[Alternate=\i]
        {Civilite MJ Std}
    e \quad}
```



올터네이트는 어떤 캐릭터에 대해 글리프를 번호 순서대로 교체해준다. `Alternate`는 숫자 0부터 시작한다. 폰트마다, 그리고 캐릭터마다 포함된 올터네이트 글리프의 개수는 다르다. 올터네이트로 대체된 글리프는 단어의 앞글자에 왔을 때, 중간에 있을 때, 마지막에 올 때 어울리는 것이 있다. 그러므로 상황과 맥락을 짚어가면서 사용해야한다. 다음 예문에서 보듯이 'Dirty dancing' 의 올터네이트 중에 네 번째와 다섯 번째는 't' 가 너무 뻗쳐있어 앞뒤 단어를 이어주는 느낌이 별로 들지 않는다. 'a' 나 'n' 의 경우도 단어 중간에는 오지 말아야할 것들이 눈에 보인다.

```
\multido{\i=0+1}{10}{%
    \fontspec[Alternate=\i,
              Color=NavyBlue]
        {Zapfino Extra LT Pro}
    Dirty dancing \quad}
```



다음 예제에 나오는 `aalt`는 `Alternate`의 오픈타입 특성 태그인데 `RawFeature`로 불러왔다. 한편, 어노테이션을 부여하면 다양한 숫자 모양을 선택할 수 있다. `Annotation`에서는 숫자가 변형된 것, 즉 박스에 둘러싸여 있거나 동그라미에 들어있는 것들만 선택되었다. 이 둘의 차이점을 비교해보자.

```
\multido{\i=0+1}{15}{%
    \fontspec[RawFeature={+aalt=\i}]
        {Adobe 명조 Std} 1234 }
```

1 2 3 4 1234 ①②③④ (1)(2)(3)(4)
🄁🄂🄃🄄 ❶❷❸❹ 🅀🅁🅂🅃 ➊➋➌➍
➊➋➌➍ ❶❷❸❹ 🄁🄂🄃🄄 ①②③④
❶❷❸❹ ➊➋➌➍ ⊢∾ↄ⊕

```
\multido{\i=0+1}{12}{%
    \fontspec[Annotation=\i]
        {Adobe 명조 Std} 1234 }
```

①②③④ (1)(2)(3)(4) 🄁🄂🄃🄄 ❶❷❸❹
❶❷❸❹ 🅀🅁🅂🅃 ➊➋➌➍ ❶❷❸❹
🄁🄂🄃🄄 ①②③④ ❶❷❸❹ ➊➋➌➍

분수의 경우 투박하지 않게 표현할 수 있고, 숫자의 모양을 문맥에 따라 위/아래 첨자로 조정해주기도 한다.

```
\fontspec{Arno Pro}
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 \\
\addfontfeature{Fractions=On}
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680
```

1/2 1/4 5/6 13579/24680

½ ¼ ⅝ 13579⁄24680

```
\fontspec[VerticalPosition=ScientificInferior]{Brioso Pro}
        H2O vs. CO2 \\
\fontspec[VerticalPosition=Ordinal]{Brioso Pro}
        21st Time
```

H_2O vs. CO_2

21^{st} time

4.1.6 세로쓰기와 다국어 처리

한국과 중국, 일본에는 세로쓰기로 된 책이 많이 있다. 이를 위해서 세로쓰기를 지원하는 오픈타입 폰트가 필요하다. XeTeX-k*o* 매뉴얼 가운데 문서의 일부분을 세로쓰기하는 데 필요한 `\vertical` 환경을 소개한다. 이 환경은 두 가지가 필요한데 하나는 세로쓰기를 지원할 글꼴 명령어, 다른 하나는 세로쓰기할 문단의 길이, 즉 줄바꿈할 세로 길이를 지정해줘야한다. 다음 예문에서는 세로쓰기할 폰트를 `\VertTypeHangul`로 지정하고 세로쓰기 문단의 길이를 판면 세로길이의 25퍼센트로 지정하였다.

```
\newfontfamily\VertTypeHangul[Mapping=tex-text, Script=Hangul,
        Language=Korean,Vertical=RotatedGlyphs]{함초롬바탕}

\begin{vertical}{\VertTypeHangul}{.25\textheight}
\begin{center}
{\large 그 사이} \\ 양희은 노래
\end{center}
\small 해 저무는 들녘 밤과 낮 그 사이로 \\ 하늘은 하늘 따라 펼쳐 널리고 \\
이만치 떨어져 바라볼 그 사이로 \\ 바람은 갈댓잎을 살 불어 가는데 \\
이리로 또 저리로 비껴가는 그 사이에 \\ 열릴듯 스쳐가는 그 사이따라 \bigskip
... (생략) ...
해 저무는 들녘 밤과 낮 그 사이에 \\ 이리로 또 저리로 비껴가는 사이에 \\
비껴가는 사이에 \\ 비껴가는 사이에
\end{vertical}
```

그 사이

양희은 노래

해 저무는 들녘 밤과 낮 그 사이로
하늘은 하늘 따라 펼쳐 널리고
이만치 떨어져 바라볼 그 사이로
바람은 갈댓잎을 살 불어 가는데
이리로 또 저리로 비껴가는 그 사이에
열릴듯 스쳐가는 그 사이따라

해 저무는 들녘 하늘가 외딴곳에
호롱불 밝히어둔 오두막 있어
노을 저 건너에 별들의 노랫소리
밤새도록 들리는 그 곳에 가려네
이리로 또 저리로 비껴가는 그 사이에
열릴듯 스쳐가는 그 사이 따라

노을 저 건너에 별들의 노랫소리
밤새도록 들리는 그 곳에 가려네
이리로 또 저리로 비껴가는 그 사이에
열릴듯 스쳐가는 그 사이 따라

해 저무는 들녘 밤과 낮 그 사이에
이리로 또 저리로 비껴가는 사이에
비껴가는 사이에
비껴가는 사이에

다음은 중국어를 가로쓰기와 세로쓰기로 조판한 예이다. 중국 명 나라 말기에 장대(張岱)가 쓴 소품문(小品文)《호심정간설(湖心亭看雪)》이다. \disablekoreanfonts는 XeTeX-k*o*에 있는 명령어인데 이를 적어주지 않으면 현재 \setmainhanjafont로 지정된 본문 한자 폰트의 한자 영역이 계속 지정되어 간체(Chinese Simplified)로 입력된 한자를 해독할 수 없다. \hanjacjksymbols는 CJK 구두점, 괄호 및 상징기호를 식자하도록 해준다. 소스 코드에도 중국어를 표현하기 위하여 모노 폰트를 잠깐 Code2000.ttf로 지정하였다.

\chinese 명령은 XeTeX-k*o*에 들어있는 명령어이다. 문단이 시작될 때 두 글자를 들여써주고 문장부호의 미세 조정이 반영된다. XeTeX-k*o*에는 일본어 조판을 위해 \japanese 명령도 마련해 놓고 있다. 텍스트는 간체로 입력되었으나 번체(Chinese Traditional) 폰트를 지정하니 간체가 번체로 대치된 것을 주목하라. 또한 별도의 조판 지정 없이 고리점(。)과 모점(、)이 세로쓰기에 어울리는 위치에 가서 붙고 큰따옴표(“”)가 겹낫표(『』)로 바뀐 것을 보라. (세로쓰기 예문 가운데 余挐一小舟 부분에서 '挐' 자가 해당 폰트에 없어 유실되었다.)

```
\newcommand\ChinSampleText{%
{\centering{\Large 湖心亭看雪}\par{\large 张岱}\par}

崇祯五年十二月、余住西湖。大雪三日、湖中人鸟声俱绝。是日更定矣、余挐一小舟、
拥毳衣炉火、独往湖心亭看雪。雾凇沆砀、天与云与山与水、上下一白。湖上影子、
惟长堤一痕、湖心亭一点、与余舟一芥、舟中人两三粒而已。

到亭上、有两人铺毡对坐、一童子烧酒、炉正沸。见余、大喜曰“湖中焉得更有此人!”
拉余同饮。余强饮三大白而别、问其姓氏、是金陵人、客此。及下船、舟子喃喃曰
“莫说相公痴、更有痴似相公者。”}
```

```
\disablekoreanfonts
\hanjacjksymbols
\fontspec[Script=CJK,Language=Chinese Simplified]{Adobe Kaiti Std}
\chinese \ChinSampleText
```

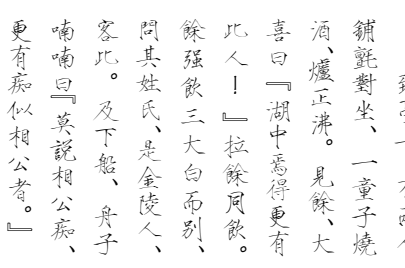
湖心亭看雪

张岱

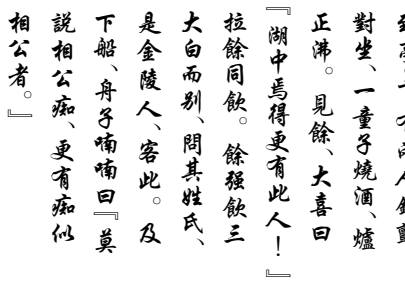
崇祯五年十二月、余住西湖。大雪三日、湖中人鸟声俱绝。是日更定矣、余挐一小舟、拥毳衣炉火、独往湖心亭看雪。雾凇沆砀、天与云与山与水、上下一白。湖上影子、惟长堤一痕、湖心亭一点、与余舟一芥、舟中人两三粒而已。

到亭上、有两人铺毡对坐、一童子烧酒、炉正沸。见余、大喜曰“湖中焉得更有此人!”拉余同饮。余强饮三大白而别、问其姓氏、是金陵人、客此。及下船、舟子喃喃曰“莫说相公痴、更有痴似相公者。”

```
\newfontfamily\ChinFontOne[%
        Script=CJK, Language=Chinese Traditional, Vertical=RotatedGlyphs]
        {FZShouJinShu-S10T}
\begin{vertical}{\ChinFontOne}{.16\textheight}
\chinese \ChinSampleText
\end{vertical}
```




```
\newfontfamily\ChinFontTwo[%
        Script=CJK, Language=Chinese Traditional, Vertical=RotatedGlyphs]
        {FZXingKai-S04T}
\begin{vertical}{\ChinFontTwo}{.15\textheight}
\ChinSampleText
\end{vertical}
```



다음은 그리스어 조판의 예이다. 호메로스(Homer)의《일리아드(Iliad)》가운데 원문 일부와 번역문을 Wikipedia에서 가져왔다. 역시 소스 코드에 그리스어를 표현하기 위하여 모노 폰트를 잠깐 `Code2000.ttf`로 지정하였다.

```
\newcommand\GreekSampleText{%
{\bfseries Homer, \emph{Iliad} IX 410--16 \\
(translated by Richmond Lattimore)}

μήτηρ γάρ τέ μέ φησι θεὰ Θέτις ἀργυρόπεζα \\
διχθαδίας κῆρας φερέμεν θανάτοιο τέλος δέ. \\
εἰ μέν κ' αὖθι μένων Τρώων πόλιν ἀμφιμάχωμαι, \\
ὥλετο μέν μοι νόστος, ἀτὰρ κλέος ἄφθιτον ἔσται \\
εἰ δέ κεν οἴκαδ' ἵκωμι φίλην ἐς πατρίδα γα□αν, \\
ὥλετό μοι κλέος ἐσθλόν, ἐπὶ δηρὸν δέ μοι αἰὼν \\
ἔσσεται, οὐδέ κέ μ' ὦκα τέλος θανάτοιο κιχείη.
\bigskip

For my mother Thetis the goddess of silver feet tells me \\
I carry two sorts of destiny toward the day of my death. \\
Either, if I stay here and fight beside the city of the Trojans, \\
my return home is gone, but my glory shall be everlasting; \\
but if I return home to the beloved land of my fathers, \\
the excellence of my glory is gone, but there will be a long life \\
left for me, and my end in death will not come to me quickly.}
```

```
\fontspec[Script=Greek,Language=Greek,Ligatures={Rare,TeX,Historic}]{Junicode}
\GreekSampleText
```

Homer, *Iliad* IX 410–16

(translated by Richmond Lattimore)

μήτηρ γάρ τέ μέ φησι θεὰ Θέτις ἀργυρόπεζα
διχθαδίας κῆρας φερέμεν θανάτοιο τέλος δέ.
εἰ μέν κ᾽ αὖθι μένων Τρώων πόλιν ἀμφιμάχωμαι,
ὤλετο μέν μοι νόστος, ἀτὰρ κλέος ἄφθιτον ἔσται
εἰ δέ κεν οἴκαδ᾽ ἵκωμι φίλην ἐς πατρίδα γαῖαν,
ὤλετό μοι κλέος ἐσθλόν, ἐπὶ δηρὸν δέ μοι αἰὼν
ἔσσεται, οὐδέ κέ μ᾽ ὦκα τέλος θανάτοιο κιχείη.

For my mother Thetis the goddess of silver feet tells me
I carry two sorts of destiny toward the day of my death.
Either, if I stay here and fight beside the city of the Trojans,
my return home is gone, but my glory shall be everlasting;
but if I return home to the beloved land of my fathers,
the excellence of my glory is gone, but there will be a long life
left for me, and my end in death will not come to me quickly.

Fontspec은 다국어 처리를 쉽게 할 수 있도록 `Script` 특성과 `Language` 특성을 정의하고 있다. 이를 이용하여 많은 언어(language)를 그에 맞는 문자 체계(script)로 표기할 수 있다. 앞의 예문에서 중국어 간체를 입력하려고 `Script=CJK`, `Language=Chinese Simlified`로 지정하였다. 이렇게 사용하고자 하는 언어와 문자처리를 위해 fontspec에 정의된 예약어를 참고한다.

4.2 맥에서만 구현되는 AAT 특성

AAT도 ICU 오픈타입의 특성과 마찬가지로 리거처, 글리프 대체, 작은 대문자, 숫자 모양, 분수 표기 등의 특성을 갖고 있다. 대부분의 AAT 특성의 옵션은 ICU 오픈타입의 특성 옵션과 대동소이하다.

4.2.1 리거처

ICU 오픈타입의 특성 외에 추가로 `Logos`, `Rebus`, `Diphthong`, `Squared`, `AbbrevSquared`, `Icelandic` 등을 지니고 있다. 리거처는 미학적인 측면이 강하므로 사용하고자 하는 폰트에 일상적인 리거처 이외에는 찾아볼 수 없을지라도 굳이 실망할 필요는 없다. 다른 특성도 마찬가지다. 글리프 대체나 특수효과는 폰트 의존적인 성격이 강하므로 잘 갖춰진 폰트를 골라쓰는 것이 좋다.

4.2.2 특수 효과

`Contextual` 특성은 ICU 오픈타입에서 소개했던 것과 다르지 않다. 옵션으로 `WordInitial`, `WordFinal`, `LineInitial`, `LineFinal`, `Inner` 등이 있고, 옵션을 해제하려면 'No'를 붙이면 된다. 여기서는 fontspec 매뉴얼의 예제를 그대로 인용한다.

```
\newfontface\fancy[Contextuals={WordInitial,WordFinal}]
        {HoeflerText-RegularItalic}
\fancy where is all the vegemite

\fontspec[Contextuals=Inner]
        {Hoefler Text}
'Inner' swashes can \emph{sometimes} \\
contain the archaic long~s.
```

where is all the vegemite

'Inner' swashes can *sometimes*
contain the archaic long s.

Variant 특성은 ICU 오픈타입의 Alternate 특성과 비슷하게 어떤 캐릭터에 대해 글리프를 번호 순서대로 교체해준다. Variant는 숫자 1부터 시작한다. 다음 예문을 ICU 오픈타입의 Alternate 예문과 비교해보자.

```
\multido{\i=1+1}{10}{%
    \fontspec[Variant=\i,
              Color=DarkRed]
        {Zapfino Extra LT Pro}
    Dirty dancing \quad}
```

Dirty dancing Dirty dancing Dirty dancing
Dirty dancing Dirty dancing
Dirty dancing Dirty dancing Dirty dancing
Dirty dancing Dirty dancing

```
\fontspec[Alternate=0]
        {Hoefler Text Italic}
Sphinx Of Black Quartz,
{\scshape Judge My Vow} \\
\fontspec[Alternate=1]
        {Hoefler Text Italic}
Sphinx Of Black Quartz,
{\scshape Judge My Vow}
```

Sphinx Of Black Quartz, JUDGE *My Vow*
Sphinx Of Black Quartz, JUDGE *My Vow*

다양한 숫자 모양을 구현하는 Annotation의 옵션으로 Box, RoundedBox, BlackCircle, Circle, DoubleCircle, BlackRoundSquare, BlackSquare, RomanNumerals, Diamond, Parenthesis, Period 등을 줄 수 있다.

```
\fontspec{Hei Regular}
1 2 3 4 5 6 7 8 9 \\
\fontspec[Annotation=Circle]{Hei Regular}
1 2 3 4 5 6 7 8 9 \\
\fontspec[Annotation=Parenthesis]{Hei Regular}
1 2 3 4 5 6 7 8 9 \\
\fontspec[Annotation=Period]{Hei Regular}
1 2 3 4 5 6 7 8 9
```

1 2 3 4 5 6 7 8 9
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
(1) (2) (3) (4) (5) (6) (7) (8) (9)
1. 2. 3. 4. 5. 6. 7. 8. 9.

특정 글리프를 본문과 어울리거나 부담스럽게 느껴지지 않도록 조판할 수 있다.

```
\fontspec{Skia} Normal
\fontspec[VerticalPosition=Superior]{Skia}
        Superior
\fontspec[VerticalPosition=Inferior]{Skia}
        Inferior \\
\fontspec[VerticalPosition=Ordinal]{Skia}
1st 2nd 3rd 4th 0th 8abcde \\
21st Century Schizoid Man
```

Normal superior inferior
1st 2nd 3rd 4th 0th 8abcde
21st Century Schizoid Man

```
\fontspec{Hiragino Maru Gothic Pro}
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 \\
\addfontfeature{Fractions=Alternate}
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680
```

1/2 1/4 5/6 13579/24680
½ ¼ ⅝ 13579/24680

4.2.3 멀티플 마스터

멀티플 마스터는 이 글의 도입부에 설명했듯이 둘 이상의 마스터 폰트로부터 다양한 모양의 폰트를 보간하여 뽑아내는 것이다. 멀티플 마스터의 폰트 특성은 AAT도 지원한다.

```
\fontspec[Weight=0.5,Width=3]{Skia}
Maybe I'll move to Mars \\
\fontspec[Weight=1,Width=1.0]{Skia}
Maybe I'll move to Mars \\
\fontspec[Weight=2,Width=0.5]{Skia}
Maybe I'll move to Mars
```

Maybe I'll move to Mars
Maybe I'll move to Mars
Maybe I'll move to Mars

5 기타

5.1 수식 폰트

Fontspec 패키지의 옵션에서 이미 설명한 바와 같이 별도의 수식 패키지를 얹지 않으면 TEX의 기본 수식 폰트인 Computer Modern (CM) 폰트를 사용한다. Fontspec은 여기에 덧붙여 수식 기호는 건드리지 않고 `\mathrm`, `\mathsf`, `\mathtt`, `\setboldmathrm`에 대해서 본문 기본 폰트(`\setmainfont`, `\setsansfont`, `\setmonofont`)에 어울리게 조정한다. 이에 더 나아가 `\mathrm`, `\mathsf`, `\mathtt`, `\setboldmathrm`을 별도로 지정할 수 있다.

```
\setmathrm[폰트 특성]{폰트 이름}
\setmathsf[폰트 특성]{폰트 이름}
\setmathtt[폰트 특성]{폰트 이름}
\setboldmathrm[폰트 특성]{폰트 이름}
```

별도의 수식 패키지를 불러오면 해당 수식 패키지에 들어있는 폰트를 사용한다. 텍에서 사용할 수 있는 수식 패키지는 *The LaTeX Font Catalogue*를 참조할 수 있다.[12]

한편, 수식 폰트를 더 잘게 나눠 지정할 수 있도록 도와주는 mathspec 패키지가 있다. Mathspec은 fonspec 패키지에 기본으로 제공하는 수식 관련 명령어 외에 `\setmathcal`, `\setmathbb`를 추가적으로 지정할 수 있다. `Digit`, `Latin`, `Greek`로 나누어 각각의 대소문자, 이탤릭 적용 여부 등을 결정할 수 있다. 또 fontspec의 `Numbers={Lining,Proportional}`, `Scale=MatchLowrcase` 등 속성 명령을 그대로 사용할 수 있다. 그러나 수식 폰트에 적합하지 않은 폰트를 수식 폰트에 사용할 경우 그리스 문자나 수식 간격이 오히려 더 나빠질 수 있다. 이때를 대비하여 `\exchangeforms`, `\normalizevarforms`, `"<문자열>`, `\"<문자열>"`, `\setminwhitespace` 명령을 준비해놓고 있다. 자세한 것은 mathspec 패키지 매뉴얼을 참고한다.

12. http://www.tug.dk/FontCatalogue/mathfonts.html

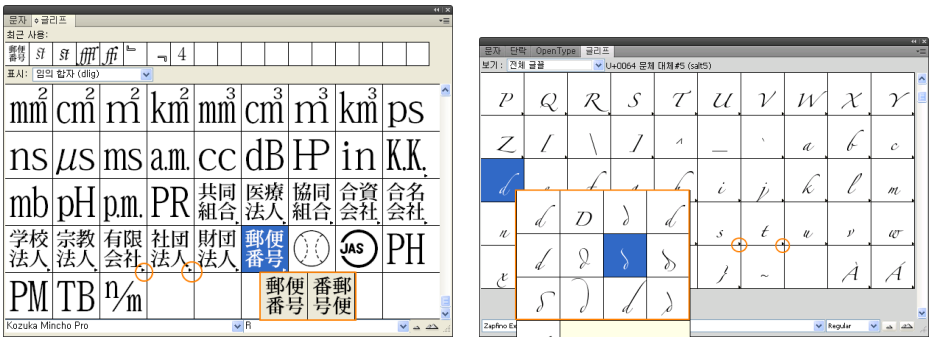


그림 10. 어도비 제품군의 글리프 팔레트

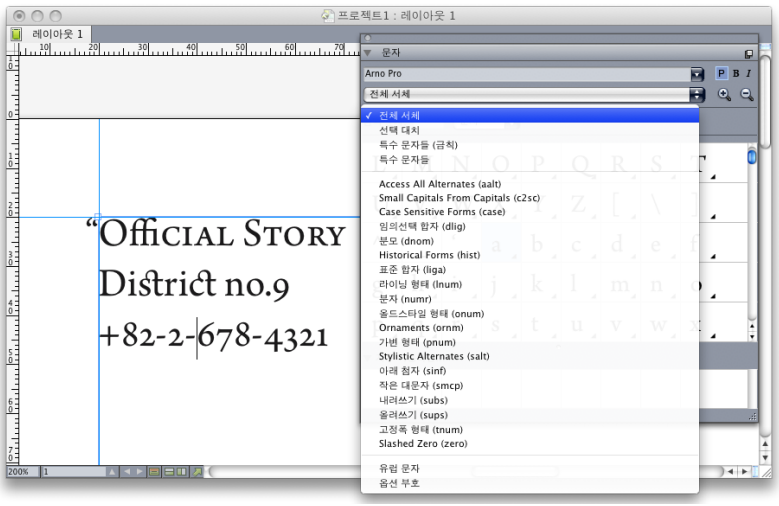


그림 11. QUARKXPRESS 8.0에서 오픈타입 폰트 특성 부여

5.2 다른 프로그램에서는?

어도비 사의 INDESIGN CS나 ILLUSTRATOR CS 등에는 그림 10과 같은 글리프 팔레트가 들어있다. 어도비 사의 제품에 들어있는 글리프 팔레트는 글리프 대체를 일목요연하게 보여준다. 글리프 팔레트에서 보여주는 어떤 캐릭터 옆에 작은 삼각형(▶)이 있으면 그에 대응하는 글리프가 들어있다는 표시이다. 이를 활용하면 다양한 글리프 대체를 구현할 수 있다. 또 오픈타입의 특성을 부여하는 별도의 메뉴가 들어있어 이를 누를 때마다 폰트 특성이 구현된다.

쿼크 사의 조판 프로그램인 QUARKXPRESS는 8.0 버전부터 본격적인 오픈타입 지원이 가능해졌다. 어도비 제품군과 비슷하게 글리프 팔레트를 제공하고 그림 11과 같이 오픈타입의 특성을 선택할 수 있는 메뉴를 제공한다.

Mac OS X 운영체제에 들어있는 문자표(CHARACTER VIEWER)는 사용자가 찍은 캐릭터를 나타낼 수 있는 모든 폰트를 보여준다. 또 해당 폰트에서 대체할 글리프가 있으면 하단에

보여준다. 또 오픈타입의 특성을 부여하는 별도의 메뉴가 들어있어 이를 누를 때마다 폰트 특성이 구현된다.

5.3 한국의 고급 글꼴 개발 현황

라틴 폰트에 비하면 늦긴 했지만 고급 특성을 담은 한글 폰트는 꾸준히 등장하고 있다. 윤디자인연구소는 봄날 글꼴을 제작하면서 가변폭과 고정폭으로 제작하였고 몇몇 글리프에 대해서는 글리프 대체를 시도할 수 있다. 국내 최초로 특성 기능을 적용한 것이라고 한다. 이외에도 폰트릭스와 산돌커뮤니케이션에서 글리프 대체 기능을 포함한 폰트를 내놓았다. (그림 12–14 참조)

한편 일부 오픈소스 커뮤니티와 기업, 관공서에서 CI를 위해 개발한 글꼴을 비상업적인 용도로 배포하는 경우도 많이 늘어났다. 글꼴의 품위는 논외로 하고 대표적인 것을 열거해보면 ko.TEX의 기본 글꼴인 은 글꼴을 필두로하여 한글과컴퓨터의 함초롬 글꼴, NHN의 나눔 글꼴, 다음커뮤니케이션의 다음체, 아모레퍼시픽의 아리따 글꼴, 조선일보의 조선일보명조, 한겨레신문의 한겨레결체, 헤움의 헤움글꼴, 서울시의 서울한강체와 서울남산체, 제주시의 제주전용서체 등 많은 폰트가 개발되어 배포되고 있다.

6 맺으며

XETEX과 fontspec 패키지의 탄생으로 텍에서 폰트 운용을 좀더 쉽게 할 수 있는 환경을 갖게 되었다. 특히 fontspec은 XELATEX을 사용한다면 없어서는 안 될 중요한 패키지로 부상하였다. 이를 사용하지 않고는 사용자의 시스템에 들어있는 각종 트루타입이나 오픈타입 폰트를 운용하는 데 조금 불편할 수 있다. 가히 fontspec을 XETEX의 날개라 부를만 하다.

앞서 한마디도 언급하지 않았지만 fontspec은 LuaTEX도 지원한다. Fontspec의 기능 가운데 LuaTEX에서만 가능한 기능, XETEX에서만 가능한 기능이 있다. 또 XETEX의 기본 드라이버인 xdvipdfmx에서만 가능한 기능, Mac OS X에만 들어있는 xdv2pdf에서만 가능한 기능도 있으므로 매뉴얼을 잘 읽어보고 상황에 맞게 사용할 것을 권장한다.

Fontspec 매뉴얼과 이 글에서 보인 많은 예제와 달리 실제로 향상된 폰트 특성을 지니고 있는 폰트는 제값을 주고 구입하지 않는 한 만나기 어려울 것이다. 어떤 폰트를 사용하는 데 매뉴얼에 소개된 폰트 특성이 제대로 구현되지 않더라도 그 폰트가 원래 그런 것이니 너무 실망하지 않아도 좋다.

한편 XETEX-k*o*를 사용하는 것은 사실상 fontspec을 사용하는 것이다. 그만큼 한글 글꼴을 운용할 수 있는 폭이 넓어졌다. 그동안 ko.TEX으로 대변되던 한글 텍 환경은 좀더 완성도 높게 변하여 XETEX-k*o*로 진화하였다. 사용자가 ko.TEX 패키지를 부르고 XETEX으로 컴파일하면 자동으로 XETEX-k*o*를 호출하고 pdfTEX으로 컴파일하면 기존 ko.TEX을 호출한다.

XETEX-k*o*를 사용하면 한글 타이포그래피를 더욱 미세하게 조정할 수 있다. 특히 ko.TEX의 `finemath` 옵션 기능과 비슷한 매크로가 들어있어 미세 간격을 조정할 수 있고, 첫가끝 입력 및 GSUB 지원 글꼴에 의한 진정한 옛한글 조판 방식을 지원하고 있다. 세로쓰기 조판과

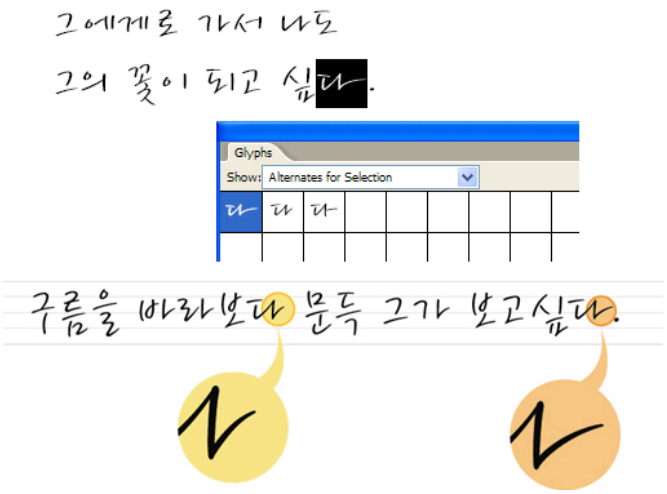


그림 12. 봄날 글꼴의 글리프 대체 (출처 : 윤디자인연구소 홈페이지)

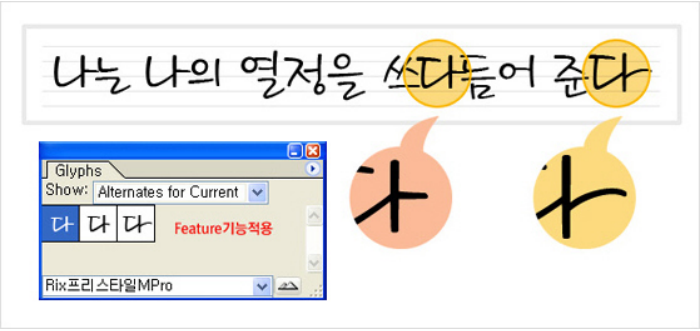


그림 13. 프리스타일 글꼴의 글리프 대체 (출처 : 폰트릭스 홈페이지)

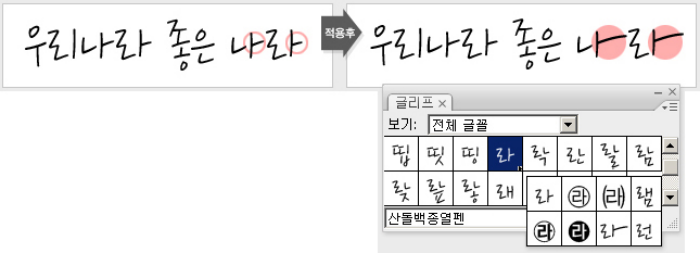


그림 14. 백종열펜 글꼴의 글리프 대체 (출처 : 산돌커뮤니케이션 홈페이지)

일본어 및 중국어 조판도 지원하며, 더 나아가 문장부호, 괄호, 라틴 알파벳 및 숫자 등을 조판할 때 라틴 폰트를 따를 것인지 한글 폰트를 따를 것인지 아니면 바로 직전 문자의 폰트를 그대로 따를 것인지로 나누어 구현할 수 있다. 자간 조절과 라틴 베이스라인에 맞춰진 문장부호의 높낮이 조절, 인용부호의 폭 조절 등 조판에서 신경써야할 많은 부분을 세밀하게 제어할 수 있다. 글쓴이는 XeTeX-k*o*가 현재 사용되고 있는 최고의 한글 조판 프로그램 중 하나라고 생각한다.

그러나 fontspec 및 XeTeX-k*o*를 사용하면서 우려되는 부분은 예전보다 글꼴을 쉽게 다룰 수 있는 탁월한 능력에 뒤따르는 폰트의 남용 문제이다. 한국의 텍 출판 여건은 좋지 않기 때문에—텍 전문 조판소나 출판사는 찾기 어렵고, 다른 조판프로그램에 비해 시장 점유율은 극히 미미한—아마 자신의 글을 출판하기 위해 XeTeX과 XeTeX-k*o*를 사용하기로 마음 먹었다면 저자가 직접 조판해야할 경우가 비일비재할 것이다. 이때 시스템에 들어있는 `.otf`나 `.ttf` 따위를 무분별하게 가져다 쓰면 출력 품질을 보장하기 어렵다. 글꼴 운용과 관련된 부분은 반드시 출판사 또는 전문가와 상의하여 검증된 글꼴을 사용하는 것을 권장한다. 전문가의 도움을 받기 어렵다면 최소한 잘 알려진 폰트를 쓰는 것이 좋다. 진수성찬을 차려놓고 거친 기장밥만 골라먹는 일은 미연에 방지해야한다.

기본적으로 XeTeX-k*o*저자의 의도를 존중하여 글꼴을 운용하고, 타이포그래피에 대한 자신이 생겼을 때 자간과 어간, 줄간격 등의 설정을 세밀하게 조정하는 것이 바람직하다. XeTeX-k*o*의 기본 설정 값은 이미 완성도를 갖추고 있다고 생각한다.

이상으로 XeTeX과 fontspec, 그리고 XeTeX-k*o*의 주요한 기능을 알아보았다. 이 글에서 언급되지 않은 내용이 있으니 각각의 매뉴얼을 정독할 것을 권한다.

마지막으로 한글 타이포그래피를 더욱 발전시키기 위해 질 높은 한글 글꼴 개발에 각계가 관심을 기울여야한다. 제대로 된 폰트가 많이 개발되어 있으면 출력물의 품질은 우선 보장할 수 있다. 최근 한글 글꼴의 경우, 예전보다 고급 속성을 가지고 있는 글꼴이 많이 늘어나기는 했지만 우리글을 막힘없이 표현하기에 충분한 완성도 높은 글꼴은 그다지 눈에 띄지 않고 있는 실정이다. 디지털 타이포그래피 시대에 제대로 된 한글 글꼴의 개발은 더욱 시급하며 이 분야의 전문가 양성도 어떤 방식으로든 획기적으로 이루어져야할 시점이라고 생각한다.

참고 문헌

1. Will Robertson and Khaled Hosny, *The fontspec Package* v2.0b, July 14, 2010. `CTAN:macros/latex/contrib/fontspec/fontspec.pdf`
   Fontspec을 사용하기로 마음 먹었다면 매뉴얼을 반드시 읽어야 한다. ICU 오픈타입과 AAT 폰트의 특성을 예제 중심으로 알기 쉽게 구성하였다.
2. 김도현, *XeTeX-ko* 간단 매뉴얼 Version 1.9, 2010. `http://ftp.ktug.or.kr/KTUG/texlive/texmf-dist/doc/xelatex/kotex-dev/xetexko/xetexko-doc.pdf`
   한글 환경에서 XeTeX과 fontspec을 사용하고 싶으면 반드시 읽어야 한다. 세밀한 한글 타이포그래피를 구현하는 매크로 설명과 예제가 들어있다.
3. Microsoft Corporation, *OpenType specification: OpenType Layout tag registry—Feature tags*, April 2002. `http://www.microsoft.com/typography/otspec/featuretags.htm`

오픈타입 폰트의 탄생 배경과 주요 기능, 기술적 측면, 특히 OpenType layout feature에 대한 정확한 정보를 제공하는 사실상 오픈타입 폰트의 홈페이지이다.

4. Adobe System Incorporated, *OpenType® User Guide for Adobe® Fonts*, October 2008. http://www.adobe.com/type/browser/pdfs/OTGuide.pdf
어도비 사는 오픈타입 폰트를 가장 많이 제작하고 있는 폰트 회사 중의 하나이다. 자신들이 만든 오픈타입 폰트가 얼마나 전문적인 타이포그래피를 구현할 수 있는지 보여준다. 어도비 폰트에 내재된 기능은 fontspec을 이용하여 구현할 수 있다.

5. David J.Perry, *Creating Scholarly Mulilingual Documents Using Unicode, Opentype, and XƎTEX*, June 21, 2009. http://scholarsfonts.net/xetextt.pdf
다국어로 이루어진 학술 문서를 조판하면서 XƎTEX과 OpenType, 그리고 fontspec이 어떻게 이용될 수 있는지 소개하였다. 문서 도입부에 TEX과 그로부터 파생된 조판 엔진, TEXworks 사용법, 다국어 조판 개념 등도 소개되어 있다.

6. Robin Williams, *The Non-Designer's Type Book*, 2nd Ed., Pearson Education Inc., publishing as Peachit Press, 2006.
(정상희 옮김, 《아름답고 프로다운 타이포그래피 101》, ㈜피어슨에듀케이션코리아, 2008)
전문적인 내용을 쉽고 간결하게, 예제 중심으로 기술한 고급 타이포그래피 입문서이다. Fontspec 매뉴얼에서 보여준 향상된 폰트 처리와 관련 예제들을 과연 어느 상황에서 어떻게 사용하는 것인지 궁금증을 풀어주기에 충분한 책이다.

폰트와 타이포그래피에 대해 더 알고 싶을 때

7. 김강수, 초간단 *xoblivoir under XƎTEX* 사용법, 2010. http://ftp.ktug.or.kr/KTUG/texlive/texmf-dist/doc/latex/kotex-dev/xoblivoir/ultrasimplexob.pdf

8. 세종대왕기념사업회 한국글꼴개발원, 《한글글꼴용어사전》, 세종대왕기념사업회, 2000.

9. 신청우, 《디지털 타이포그래피》, 임프레스, 2003.

10. 조진환, TEX: 조판, 그 이상의 가능성, The Asian Journal of TEX **1** (2007), no. 1, 3–16. http://ajt.ktug.kr/2007/0101chof1.pdf

11. Adobe System Incorporated, *Adobe Type 1 Font Format*, Addison-Wesley Publishing Company, Inc., February 1993. http://partners.adobe.com/public/developer/en/font/T1_SPEC.PDF

12. Adobe System Incorporated, *PostScript® Language Reference*, Third Edition, Addison-Wesley Publishing Company, February 1999. http://www.adobe.com/products/postscript/pdfs/PLRM.pdf

13. Adobe System Incorporated, *Typography Primer*, 2000. http://www.adobe.com/education/pdf/type_primer.pdf

14. Apple Computer Inc., *AAT Font Feature Registry*, Apple Inc., 1998. http://developer.apple.com/fonts/registry

15. Apple Computer Inc., *Advanced Typography with Mac OS X Tiger: Using and Managing Fonts*, October 2004. http://images.apple.com/pro/pdf/L311277A_FontTT_v4.pdf

16. Robert Bringhurst, *The Elements of Typographic Style*, version. 3.1, Hartley & Marks, 2005.

17. Michel Goossens, *XƎTEX Companion: TEX meets OpenType and Unicode*, January 11,2010. http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf

18. Yannis Haralambous, *Fonts & Encodings: From Unicode to Advanced Typography and Evreything in Between*, O'Reilly Media, Inc., 2007.

19. Gareth Hughes, *Free Unicode fonts: Latin script*, August 29, 2009. http://www.garzo.co.uk/documents/freefonts.pdf

20. Jonathan Kew, "XeTeX: the Multilingual Lion — TEX meets Unicode and smart fonts," in TUG 2005 Conference, Wuhan, China, August 2005. http://www.tug.org/mactex/src/Demos/XeTeX-showcase/xetex-wuhan.pdf

21. Werner Lemberg, *Unicode Support in the CJK Package*, The Asian Journal of TEX **2** (2008), no. 1, 11–20. http://ajt.ktug.kr/2008/0201lemberg.pdf

22. Andrew Gilbert Moschou, *The mathspec package: Font selection for mathematics with XeLaTeX*, September 30, 2009. CTAN:macros/xetex/latex/mathspec/mathspec.pdf

23. Thomas W. Phinney, *TrueType, PostScript, Type 1 & OpenType: What's the Difference?*, Version 2.36, 2004. http://blogs.adobe.com/typblography/TT%20PS%20OpenType.pdf

24. Will Robertson, *The XeTeX reference guide*, July 12, 2010. CTAN:info/xetexref/XeTeX-reference.pdf

25. Ilene Strizver, *Type Rules!: the designer's guide to professional typography*, second edtion, John Wiley & Sons Inc., 2006.
(선병일 · 이지현 · 이재선 옮김, 《디자이너가 꼭 알아야 할 타이포그래피》, 디자인코리아, 2009)

26. Keith Chi-hang Tam, *Digital Typography: a primer*, 2006. http://www.keithtam.net/documents/keithtam_digital_type_primer.pdf

27. Jürgen Willrodt, "OpenType Status 2009," in DTL FontMaster Conference *Type[&]Design 2009*, November 2009. http://www.fonttools.org/downloads/TD_2009/OpenType_Status_2009.pdf

28. Wikipedia, 폰트 및 타이포그래피와 관련한 많은 항목들, 2010. http://www.wikipedia.org