http://dx.doi.org/10.5392/JKCA.2011.11.12.560

안드로이드에서 힌디어 텍스트 처리 방법

A Text Processing Method for Devanagari Scripts in Andriod

김재혁, 맹승렬
공주대학교 컴퓨터공학전공

Jae-Hyeok Kim(widedev@kongju.ac.kr), Seung-Ryol Maeng(srmaeng@kongju.ac.kr)

요약

본 논문에서는 개방형 OS인 안드로이드에서 힌디어 텍스트 처리방법을 제안한다. 텍스트 처리의 핵심은 알파벳을 문자로 조합하는 규칙을 정의하는 오토마타와 폰트 파일에서 문자에 대응하는 이미지를 검색하고 이를 화면에 표시하는 폰트 렌더링이다. 오토마타는 입력 문자의 종류와 개수에 좌우되는데 유니코드를 기반으로 자음 14자와 모음 34자를 알파벳으로 사용하는 오토마타를 제안한다. 조합된 음절은 테이블 매핑 방식을 사용하여 그립 인덱스로 변환하고 해당하는 폰트를 로드하기 위한 핸들로 사용한다. 프리 타입 폰트엔진의 다국어 지원 프레임워크에 따라 제안방법을 별도의 모듈로 추가함으로서 시스템 수준에서 힌디어를 지원할 수 있다. 메시지 어플리케이션을 통해 제안방법의 타당성을 보인다.

■ 중심어 : | 힌디어 | 텍스트 처리 | 오토마타 | 그립 | 폰트엔진 |

Abstract

In this paper, we propose a text processing method for Hindi characters, Devanagari scripts, in the Android. The key points of the text processing are to device automata, which define the combining rules of alphabets into a set of syllables, and to implement a font rendering engine, which retrieves and displays the glyph images corresponding to specific characters. In general, an automaton depends on the type and the number of characters. For the soft-keyboard, we designed the automata with 14 consonants and 34 vowels based on Unicode. Finally, a combined syllable is converted into a glyph index using the mapping table, used as a handle to load its glyph image. According to the multi-lingual framework of Freetype font engine, Dvanagari scripts can be supported in the system level by appending the implementation of our method to the font engine as the Hindi module. The proposed method is verified through a simple message system.

■ keyword : | Hindi Language | Text Processing | Automata | Glyph | Font Engine |

Ⅰ. 개 요

문자는 컴퓨터가 처리해야할 기본 정보이며 이를 입력하는 기능은 모든 컴퓨터의 필수기능이다. 이런 점에서 스마트폰의 환경에 적합한 다국어 문자입력 방법이 필요하고 스마트폰의 다양한 응용 프로그램을 활용할 수 있다. 그러나 스마트폰은 별도의 키보드를 갖는 대신 터치스크린에 문자 키보드를 표시하고 이를 접촉하

접수번호 : #110824-004
접수일자 : 2011년 08월 24일
심사완료일 : 2011년 09월 28일
교신저자 : 맹승렬, e-mail : srmaeng@kongju.ac.kr

여 문자를 입력하는 소프트 키보드 방법을 사용하기 때문에 많은 수의 문자를 배열할 수 없고 컴퓨터용 문자 입력방법을 그대로 사용할 수 없다. 예를 들면, 한글은 자소단위의 조합방식을 사용하여 키의 숫자를 최소한으로 줄여서 사용하는데, 힌디어도 조합구조를 갖기 때문에 동일한 문제가 발생한다.

스마트폰은 애플이 주도하는 iOS 기반의 iPhone과 안드로이드 기반의 안드로이드 폰으로 나눌 수 있는데 모두 소프트 키보드를 개발할 수 있는 환경을 제공한다. iOS는 공급자 중심의 운영체제로서 다양한 응용 프로그램을 개발할 수 있도록 안정된 개발환경을 제공하기 때문에 다양한 문자입력기를 쉽게 개발할 수 있고, 현재 50여개 문자입력기가 탑재되어 시판되고 있다. 반면 개방형 운영체제인 안드로이드는 커널과 프로그램 실행환경만 제공하고 제품의 구현은 개발자에게 맡겨져 있다[4]. 기본적으로 다국어 문자를 지원하도록 설계되어 있지만 언어의 특성에 기인된 문제는 별도로 개발하여야 하기 때문에 스마트폰 제조사별로 문자입력기가 다를 수 있다. 스마트폰 시장은 애플이 주도하고 있으나 삼성, LG 등 국내 업체도 안드로이드 기반의 스마트폰을 개발하여 세계 시장에 진입하고 있다[2]. 특히 동남아 시장이 점차 커지고 있기 때문에 이들 나라의 문자처리 S/W 개발이 필요하다.

지금까지 다국어처리에 대한 연구는 주로 PC용 문서편집기나 DB에서 다국어 처리를 대상으로 한다[8][9]. 핸드폰에서 다국어 처리도 많은 연구결과가 있는데, 주로 한글, 한자, 일본어를 대상으로 한다[3][5][6][10]. 그러나 아직 안드로이드 플랫폼에서 다국어 지원에 대한 연구는 많지 않고, 스마트폰 제조회사를 중심으로 한글에 대해서만 개발이 이루어진 수준이고, 특히 힌디어에 대해서는 기존 연구결과를 찾을 수 없다.

일반적으로, 문자처리는 구문을 분석하는 오토마타와 그립 검색과 렌더링을 담당하는 폰트엔진으로 구성된다. 안드로이드는 다국어 지원을 위해 가변길이 문자코딩방법인 유니코드 UTF-8[11]을 지원하는데 구문분석기는 UTF-8로 표현된 알파벳을 인식하며, 그립 검색기는 문자에 해당하는 그립 이미지 정보를 읽어오며 문자코드를 그립의 인덱스로 변환하여 실행한다. 그리고 렌더링은 검색된 그립의 기하학적 정보를 이용하여 그리기는 과정이다.

산스크리트어를 어원으로 하는 힌디어는 한글과 마찬가지로 조합형 구조를 가진다[7]. 가장 큰 특징은 자음과 모음이 중첩될 수 있고, 모음과 결합되는 과정에서 자모음 모양이 바뀔 수 있다는 점이다. 한글 조합형 문자처리에서도 경험한 바와 같이 완전 조합형은 코드화된 자소의 개수가 적어 스마트폰의 자판 설계에 적당한 반면 음절이 완성되어 가는 과정에서 자소의 모양이 달라지므로 복잡한 처리과정이 필요하다. 문제의 핵심은 자소를 음절로 조합하는 과정에서 문자코드를 그립 인덱스로 매핑하는 방법인데, 보통 문자와 그립은 1대1 대응관계가 성립하지 않고, 힌디어는 그립의 변화가 다양하므로 깊은 고려가 필요하다.

본 논문에서는 안드로이드에서 힌디어를 지원하기 위한 방법을 제시한다. 59개의 기본자소를 사용하여 음절을 조합하는 오토마타를 설계하고, 조합된 음절의 그립을 렌더링하는 방법을 제안한다. 시스템 수준에서 힌디어를 처리하기 위해 Freetype[13] 폰트 렌더링 라이브러리의 구조를 분석하고 방향을 제시한다.

본 논문은 5장으로 구성되며, 2장에서는 안드로이드의 폰트처리 구조의 특징과 이를 이용한 다국어 문자입력기 구현의 문제점을 도출하고, 3장에서는 힌디어 조합 오토마타와 그립 매핑방법을 설명하고, 4장에서는 힌디어 문자 입력기 구현결과를 제시한다. 그리고 마지막으로 5장에서 본 논문의 의미를 살펴본다.

II. 안드로이드의 다국어처리

1. 안드로이드의 다국어 지원

안드로이드는 리눅스를 기반으로 개발된 스마트폰용 운영체제로, [그림 1]에서 보는 바와 같이 커널, 라이브러리, 어플리케이션 프레임워크 3계층으로 이루어져 있으며, 응용프로그램은 이들 위에서 실행된다[1][12]. 리눅스 커널은 운영체제가 기본적으로 제공해야 할 보안, 메모리 관리, 프로세스 관리, 네트워크 스택, 드라이버 모델과 같은 핵심 시스템 서비스를 제공하며, 그 위에

는 그래픽 기능을 제공하는 OpenGL ES, 데이터베이스를 제공하는 SQLite, 폰트 렌더링 기능을 제공하는 Freetype 등이 위치한다. 또한 자바 가상머신과 같이 응용 프로그램 실행을 직접 지원하는 안드로이드 런타임 환경도 이 계층에서 지원한다.

어플리케이션 프레임워크 API는 하위 라이브러리 함수로 만들어진 툴킷으로, 개발자들이 보다 쉽게 고급 기능들을 사용할 수 있도록 설계되어 있다. 따라서 어플리케이션 개발자는 보통 어플리케이션 프레임워크가 제공하는 API(Application Programmer Interface)를 통하여 필요한 라이브러리에 접근한다.

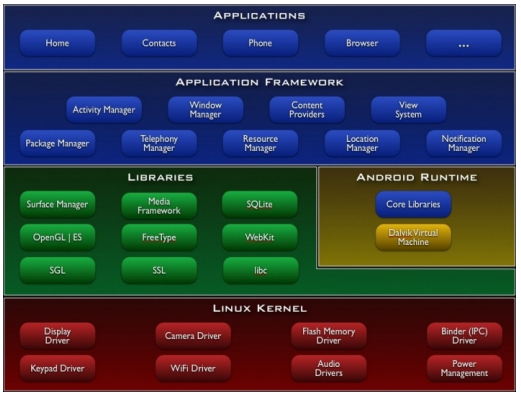
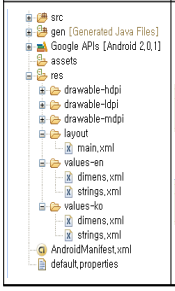


그림 1. 안드로이드 시스템 구조

안드로이드는 UTF-8 유니코드[11]를 사용하여 기본적으로 시스템 수준에서 다국어 문자 처리를 지원한다. 이러한 이유로 안드로이드 폰에서는 간단하게 다국어 사용 환경을 설정할 수 있다. 먼저 안드로이드 폰에 대한 루트 접근권한을 얻어 특정 언어의 폰트를 설치한다. [표 1]과 같이 응용 프로그램의 프로젝트를 생성하고, res/values 폴더를 언어별로 세분화하여 dimens.xml 파일과 string.xml 파일을 코딩하면 원하는 언어를 사용할 수 있다. 여기서 안드로이드가 다국어를 지원한다는 것은 Freetype 폰트 엔진에서 해당 언어를 지원한다는 의미이나, 한글이나 힌디어와 같은 일부 언어는 직접 지원하지 않기 때문에 다국어 처리를 위한 추가적인 모듈이 필요하고 이에 대한 연구가 필요하다.

표 1. 다국어 사용 방법

프로젝트 구성	value 폴더 세분화	결과
		

2. 안드로이드의 폰트 엔진

우리가 관심을 갖는 다국어처리와 직접 관련된 부분은 입력된 문자의 이미지를 지정된 위치에 표시하는 과정, 즉 폰트처리 부분이다. 안드로이드의 폰트엔진 Freetype은 트루타입, Type1, Opentype 등 다양한 포맷의 폰트 파일에 접근하여 그립(glyph) 정보를 추출하는 저수준의 라이브러리이다. 텍스트를 다양한 형태로 출력하거나, 정확한 폰트의 치수(metric)를 계산하거나, 플랫폼에 독립적인 텍스트 관리 툴을 만들고자 할 때 유용한 저수준 라이브러리이며, 공개 소프트웨어로서 쉽게 사용자화 할 수 있도록 설계되어 있다.

표 2. Freetype 폰트파일 구조

구분	설명	예
glyph	문자의 모양	AaBbCc...
Character Map	문자코드와 그립 ID 맵	(41,0),(61,1)
Metrics	그립의 가로, 세로 치수	A의 폭은 36 a의 폭은 24
Hints	그립 이미지의 그림 모양	H의 두 세로획은 같은 두께로
Kerning	문자간 그리기 효과	A와 V의 간격은 좁게

MS Window에서 OpenGL이 텍스트처리를 위해 Window GDI를 사용하듯이 안드로이드 2D 그래픽 라이브러리 SGL도 텍스트 처리를 위해 Freetype을 사용한다. 텍스트 출력이나 텍스트 레이아웃과 같이 고급기능을 지원하지 않지만 필요에 따라 고수준 함수를 만들 수 있는 환경을 제공하기 때문에, Skia 사는 Freetype

라이브러리를 기반으로 자신의 고수준 텍스트 출력 라이브러리를 개발하여 제공한다.

Freetype의 폰트 파일은 [표 1]과 같이 간단한 구조를 갖는다. 그립은 글자의 모양을 나타내는 외곽선 정보, 즉 라인 세그먼트와 Bezier 곡선의 제어점 정보를 포함하며, 비트맵 폰트를 사용할 경우는 외곽선 폰트를 비트맵으로 변환하여 사용할 수 있도록 함수를 지원한다. 문자맵(character map)은 어플리케이션에서 전달되는 문자 코드를 해당 그립으로 변환하기 위한 맵이다. 치수(metrics)는 각 그립의 폰트에 대한 치수정보를 포함한다. 힌트와 커닝정보는 가독성을 높이기 위해 텍스트의 모양을 꾸며서 정보를 포함하고 있다.

기본적으로, 폰트 렌더러의 기능은 폰트 파일로부터 그립(glyph) 이미지와 치수 정보를 읽어 지정된 위치에 그려주는 것으로, [표 3]은 Freetype의 텍스트처리 알고리즘을 보여준다.

표 3. 폰트 렌더링 알고리즘

1. 문자열의 한 문자에 대해 문자코드를 그립 인덱스로 변환한다.
2. 펜의 위치를 커서 위치에 고정한다.
3. 폰트 파일로부터 그립 이미지를 로드한다.
4. 폰트 이미지의 원점이 커서 위치와 일치되도록 변환한다.
5. 그립 이미지를 장치에 표시한다.
6. 픽셀 단위로 그립 끝까지 펜 위치를 증가시킨다.
7. 문자열의 모든 그립 이미지가 그려질 때까지 스탭 3~5까지 반복한다.
8. 모든 그립이 그려지면 텍스트 커서를 새로운 위치로 이동한다.

영어의 경우 모든 알파벳이 유니코드로 표현되기 때문에 알파벳의 그립 이미지만 있으면 문자 코드가 그립 인덱스로 쉽게 변환되며 [표 3]의 알고리즘을 적용하여 쉽게 폰트를 렌더링할 수 있다. 그러나 힌디어는 UTF-8에 기본 자소만 배치되어 있어 조합과정에서 생성되는 그립에 대해 문자코드가 존재하지 않기 때문에 문자 코드를 그립 인덱스로 직접 매핑할 수 없고, [표 3]의 폰트 렌더링을 그대로 적용할 수 없다. 따라서 힌디어를 처리하기 위해서는 코드화된 기본 자소 문자를 조합하는 규칙과 이를 그립 인덱스로 매핑하는 기능을 폰트 엔진에 추가해야 한다. 다음 장에서는 이에 대해 설명한다.

III. 안드로이드에서 힌디어 처리 방법

1. 시스템 수준 접근 방법

[그림 2]에서 보는바와 같이 다국어를 지원하는 방법은 두 가지 접근방법이 있다. 하나는 어플리케이션 수준의 방법이고, 다른 하나는 시스템 수준에서 지원하는 방법이다. 전자는 개발자가 어플리케이션을 개발할 때, 전용 API를 사용하기 때문에 자신의 어플리케이션에서 특정 언어의 지원이 가능하지만 제 3자가 개발한 어플리케이션에서는 동일한 문자가 지원될 수 없는 단점이 있다. 반면 후자는 시스템을 사용자화해야 하지만 모든 어플리케이션에 일관된 방법으로 다국어를 지원할 수 있는 장점이 있다.

폰트 렌더링 알고리즘에서 언어적 특성에 따라 달라지는 부분은 [표 3]의 1단계, 즉 문자 코드를 그립 인덱스로 변환하는 과정이다. 이 과정을 시스템 수준에서 구현하기 위해서는 Freetype 엔진을 사용자화해야 하는데, 이는 시스템에 접근하는 작업이기 때문에 장치 개발자가 담당한다.

일반적으로, 시스템 라이브러리를 사용자화하는 방법은 시스템 라이브러리와 연동되는 정적 라이브러리를 개발하는 것이다. 마찬가지로 힌디어를 처리하기 위해 조합규칙과 그립 매핑 기능을 정적 라이브러리로 개발하고 Freetype과 연동시킨다. 그렇게 함으로서 기존의 다국어 처리와 충돌을 막을 수 있다. 안드로이드 폰에서 한글의 경우도 [그림 3]과 같이 한글 조합규칙과 그립 인덱스 변환을 위해 같은 접근방법을 사용한다.

다음 절에서는 힌디어 지원기능의 핵심인 힌디어 조합 오토마타를 중점적으로 다룬다.

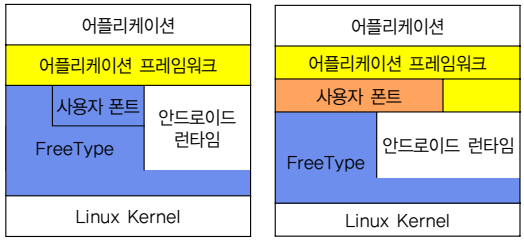


그림 2. 다국어 지원 방법((a) 응용프로그램 수준 (b)시스템 수준)

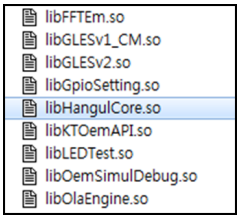


그림 3. 안드로이드에서 한글 지원

2. 힌디어 오토마타

본 절에서는 힌디어의 자소가 부호화 문자열로 입력될 때, 이들을 구문분석(lexical analysis)하여 음절로 조합하기 위한 조합 오토마타를 정의한다.

힌디어의 어원은 산스크리트어로 한글과 같이 기본 자소가 음절로 조합되는 원리를 가지고 있다. 힌디어의 쓰기 시스템의 기본형은 (식 1)과 같이 단순하다:

CV. (식 1)

여기서 C는 자음, V는 모음을 나타낸다.

그러나 중자음과 복자음을 허용하기 때문에 힌디어의 일반형은 (식 2)와 같이 복잡한 형식을 가진다:

(((C)C)C)((V)V). (식 2)

힌디어는 종성이 없으며, 음절의 구성은 (식 2)를 (식 1)로 단축해가는 과정이라고 할 수 있다. 자음을 예로 들면, (식 2)에서 보는 바와 같이 힌디어는 3개까지의 복자음이 모음과 결합할 수 있다. 따라서 복자음 क्क = क्(반자음)+क(자음) = क(ka)+्(virama)+क(ka) 이 가능하다. 결과적으로 하나의 복자음은 3~4개까지의 문자코드가 결합되고 이 결합 코드를 완성된 복자음 그립 인덱스로 변환하는 것이 필요하다. 또한 복자음 + 모음이 결합되어 음절의 그립을 나타낼 때 더 많은 결합코드가 생성되고, 나아가 क+आ=का 와 같이 해당 음절의 그립 모양이 완전히 달라지는 경우도 있다. 윈도우 7은 복자음 조합 규칙을 가지고 있어 대부분의 음절을 렌더링하지만, 국내 상용 안드로이드에서는 유니코드에 등록된 문자에 대해 단모음 음절 CV 형식에 대해서만 렌더링을 지원한다. 따라서, 복자음 क+्+क = क्क 이 아닌 क् क 로 출력된다.

힌디어 조합 오토마타를 H라 할 때, H는 (식 3)과 같이 5개 항의 튜플로 정의된다.

H = (Q, Σ, δ, q0, A), (식 3)

여기서, Q는 상태들의 유한집합, Σ는 힌디어 알파벳, δ는 상태천이 함수, q0는 시작 상태, A는 인식상태를 나타낸다.

제안하는 힌디어 오토마타는 총 8개의 상태집합 Q={q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7}으로 이루어지고, 최종적으로 힌디어는 상태 A={q_4, q_7}에서 인식된다. 제안하는 오토마타에서 사용하는 알파벳은 UTF-8에 등록된 힌디어 자음과 모음 중에서 총 59자를 사용하고 기능에 따라 [표 4]과 같이 분류한다. 모음은 기본 모음은 11자, 유성모음 2자과 비음모음 2자를 사용하고, 자음은 기본 자음은 33자, 반자음 기호 1자를 사용한다. 힌디어 음절 조합을 위한 상태천이 함수는 [표 5]와 같으며, 상태천이 함수는 δ(현재상태, 입력, 다음상태)의 형식으로 표현한다. [그림 4]는 힌디어 인식 상태천이도를 보여주며, 이 오토마타는 총 15,544개의 힌디어 음절을 인식한다.

표 4. 힌디어 자음 모음

구분		알파벳										
모음 (V)	V_b	अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ
	V_x	◌	◌ः	◌ँ	◌ं							
자음 (C)	C_b	च	छ	झ	ञ	ट	ठ	ड	ण	त	थ	द
		ध	न	प	ब	भ	म	ल	ळ	व	श	ष
		स										
	C_{bb}	क	ख	ग	घ	ज	फ	य	र	ड	ढ	
	C_x	◌्										

표 5. 상태 천이 함수

현재상태	천이함수	다음상태
q_0	$\delta_1(q_0, \in V, q_1)$	q_1
q_0	$\delta_2(q_0, \in C, q2)$	q_2
q_2	$\delta_3(q_2, c \in C, q_0)$	q_0
q_0	$\delta_4(q_0, c \in C, q_3)$	q_3
q_1	$\delta_5(q_1, c \in C_{bb}, q_3)$	q_3
q_3	$\delta_6(q_3, \in C, q_4)$	q_4
q_3	$\delta_7(q_3, \in C, q_5)$	q_5
q_3	$\delta_8(q_3, v \in V, q_6)$	q_6
q_3	$\delta_9(q_3, \in V, q_7)$	q_7
q_5	$\delta_{10}(q_5, v \in V, q_6)$	q_6
q_7	$\delta_{11}(q_7, v \in V, q_6)$	q_6

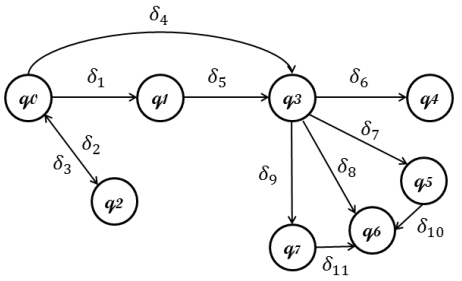


그림 4. 힌디어 오토마타의 상태천이 다이어그램

오토마타를 통해 인식된 문자는 그립 인덱스로 변환되어 후속 폰트 렌더링 과정을 거치게 되는데, 이는 다음 절에서 설명한다.

3. 힌디어 폰트엔진 사용자화

오토마타를 통해 인식된 문자를 출력하기 위해서는 폰트를 파일로부터 로드해야 하는데 이 때 그립 인덱스가 사용된다. 이 후, 폰트 렌더러는 그립의 인덱스를 중심으로 문자열을 처리한다. 영어나 한글의 완성형 코드처럼 문자의 코드와 그립의 인덱스가 1대1 대응되는 경우는 문제가 없다. 그러나 힌디어와 같이 문자코드와 그립 인덱스가 대응되지 않는 경우는 폰트 엔진에서 그립 매핑문제를 처리해야 한다.

제 2장에서 살펴본 바와 같이, Freetype 의 그립 렌더링은 크게 3단계에 의해 이루어지며, 각각의 단계에 대해 [표 6]와 같은 API를 제공한다. 여기서 실제 다국어 처리와 관계가 되는 API는 FT_Get_Char_Index()이기 때문에 이 함수에서 오토마타와 그립 매핑문제에 대한 논의가 필요하다. 실제로 그립 이미지의 로드나 그리기 작업은 변경할 필요가 없다. 또한 그립 이미지의 저장형태는 그리 중요하지 않다. Opera 웹 브라우저는 기본 그립 이미지는 자체에 내장하고 있지만 복합문자는 필요한 경우 서버로부터 다운받는 방법을 사용한다. 어떤 장치들은 메인 메모리에 폰트파일을 저장하거나 외장 디스크에 폰트의 그립 이미지를 저장하는 것도 가능하다.

표 6. Freetype 폰트엔진의 주요 API

API 명	기능
FT_Get_Char_Index (face, TEXT("string"))	문자코드를 그립 인덱스로 변환
FT_Load_Glyph(face,Index, FT_LOAD_DEFAULT|FT_LOAD_NO_BITMAP)	그립 이미지 로드
FT_Render_Glyph (face,glyph,FT_RENDER_MODE_NORMAL)	그립 이미지 그리기

FT_Get_Char_Index() 함수에서 매개변수로 사용된 문자열의 처리는 변경이 불가피 하다. 이 함수는 힌디어 자모만 UTF-8로 표현되어 있으므로 복자음이나 완성된 음절에 대해서는 코드화가 되어 있지 않기 때문에 매개변수 스트링을 직접 그립 인덱스로 변환할 수 없다. 따라서 Freetype 라이브러리를 기반으로 다국어를 지원하기 위해서는 FT_Get_Char_Index()가 힌디어 오토마타를 호출하고, 각 오토마타 상태에 대응되는 그립 인덱스를 계산하도록 수정되어야 한다.

표 7. FT_Get_Char_Glyph

```
void FT_Get_Char_Glyph(face, char string[]) {
    ......
    int glyphIndex;
    extern static int currentLetter; //현재 자모음 위치
    extern static int state;

    while (string[currentLetter] !=  "\n" ) {
     glyphIndex=hAutomata(string[i], state);
    }
}
bool hAutomaton(string s, int state) {
    .........
    int index; int newState
    newState=automata(state,s);
    for newState
        index=charToglyph(newState);
        return(index);
}
```

[표 7]은 UTF-8 바이트 스트림으로 전달되는 힌디어 문자열을 조합하고 그립 인덱스로 매핑하기 위해 FT_Get_Char_Glyph()함수를 사용자화하기 위한 골격을 보여준다. hAutomata()함수는 현재 오토마타 상태에서 입력되는 하나의 자소 문자열에 대해 오토마타를

실행하는 함수로, 실행결과로 조합된 새로운 상태를 리턴한다. 그런데 문자가 조합되는 과정에서 중간 문자에 대한 그립은 화면에 계속적으로 보여져야하기 때문에 새로운 상태에서는 조합된 문자에 해당하는 그립 인덱스를 계속적으로 계산해야 한다.

본 논문에서는 힌디어 조합에 과정에서 생성되는 문자의 그립에 대해 테이블 기반 그립 인덱스 매핑방법을 사용한다. 그립 인덱스의 매핑은 폰트 파일의 구현방법에 종속적이므로, 먼저 힌디어 폰트의 구성에 대해 살펴본다. 힌디어는 음절의 개수가 많기 때문에 보통 조합형 폰트를 사용한다. 조합형 폰트는 모든 음절에 대해 그립을 디자인하지 않고, 자음과 모음의 그립만 디자인하여 조합하는 방식이다. 따라서 그립 매핑은 자음과 모음 각각에 대한 매핑만 생각하면 된다. 오토마타의 자음상태 q_0, q_1, q_2, q_3 에서 자음 그립을, 모음상태 q_5, q_6에서 모음 그립을 각각 매핑한다. 1[표 8]은 자음과 모음 그립이 조합되어 음절을 표시하는 예를 보여준다.

표 8. 힌디어 기본 자모의 조합

자음 / 모음	अ(a)	आ(aa)	इ(i)	ई(ii)	उ(u)	ऊ(u)	ऋ(ri)
क(ka)	क	का	कि	की	कु	कू	कृ
क़	क़	क़ा	क़ि	क़ी	क़ु	क़ू	क़ृ
ख(kha)	ख	खा	खि	खी	खु	खू	खृ
ख़	ख़	ख़ा	ख़ि	ख़ी	ख़ु	ख़ू	ख़ृ
ग(ga)	ग	गा	गि	गी	गु	गू	गृ
ग़	ग़	ग़ा	ग़ि	ग़ी	ग़ु	ग़ू	ग़ृ
घ(gha)	घ	घा	घि	घी	घु	घू	घृ

Ⅳ. 구현 및 실험

1. 소프트키보드에서 힌디어 입력

스마트폰은 휴대하기 쉽게 설계되어야 하기 때문에 입력 장치가 컴퓨터에 비해 상당히 빈약하다. 이것을 보안하기 위해 안드로이드에서는 소프트키보드 환경을 제공한다. 소프트키보드란 터치 화면상에 화상 키보드를 표시해 입력을 받는 소프트웨어로 구현된 자판이다. 소프트키보드는 하드웨어인 일반 키보드와 달리 소프트웨어로 만들어졌기 때문에 다양한 언어와 다양한 자판배열을 가진 키보드를 사용자가 원하는 대로 바꿔가며 사용할 수 있다는 이점이 있다.

안드로이드 개발 키트의 샘플코드에 있는 Sample Softkeyboard라는 소프트키보드를 통해 소프트키보드의 구조를 알아볼 수 있다. [그림 5]에서 보는 바와 같이 Softkeyboard는 다른 안드로이드 어플리케이션과 마찬가지로 xml 소스, java 소스, 이미지들로 구성되어있다. XML 소스에는 키보드의 레이아웃과 키 코드가 정의되어 있다. XML 소스는 어플리케이션이 실행될 때 키보드 생성을 위한 데이터로 사용된다. JAVA 소스는 키보드 어플리케이션을 동작시키는 실질적인 소스로 키보드를 생성하여 키 이벤트를 처리한 후 키보드 입력을 요청했던 어플리케이션으로 문자열을 전송하는 역할을 한다. 이미지는 줄 바꿈, 공백, 검색 기호 등 문자로 표현하기 어려운 자판을 출력하기 위해 보조적으로 사용된다.

소프트키보드는 텍스트를 입력하기 위해 키보드가 필요하다는 요청을 받으면 먼저 XML 소스에 있는 레이아웃 데이터를 읽어와 화면에 키보드를 표시한다. 그 후 버튼이 눌리면 수시로 키 이벤트가 발생하여 미리 정의한 키 코드에 따라 문자를 처리한다. 이때, 처리된 문자는 작성중인 문자열에 추가되며 작성이 종료되면 (Commit), 키보드를 요청한 어플리케이션에 문자열을 보내고 버퍼를 비우게 된다. 그러나 작성이 끝나지 않았더라도 작성중인 문자를 계속 출력해줄 필요가 있기 때문에 문자를 입력 받을 때마다 작성중인 문자열을 해당 어플리케이션에 보내게 된다. 작성중인 문자열이라는 것은 안드로이드에서는 밑줄로 표현하고 있다. 입력이 종료되기 전까지는 작성중인 문자열을 키보드 어플리케이션이 계속 가지고 있기 때문에 작성중인 문자열의 맞춤법 검사나 자동 대문자와 같은 부가적인 처리를 할 수 있다.

1. 자료를 제공해 주신 (주)KPA 김의철 대표께 감사드립니다.

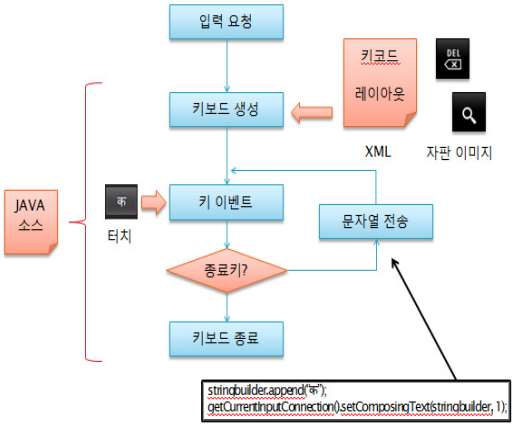


그림 5. 소프트키보드 소스 구조

소프트키보드는 이런 간단한 구조로 되어 있어 개발하기 편리하다는 이점이 있지만 분명한 한계가 있다. 사용하기 쉬운 고수준의 함수로 구성되어 있어 입력한 문자 데이터를 입력 요청 어플리케이션으로 전송하는 것은 문자열 형태로 밖에 할 수 없다. 따라서 코드로 직접 전송하거나 다른 데이터 형식으로 보내는 세부적인 방법은 사용할 수 없다. 또한 키를 입력 받아 문자열로 보내는 기능만을 하기 때문에 문자 출력을 제어할 수 있는 권한은 오로지 키보드를 요청한 어플리케이션에 있다. 따라서 다국어를 입력할 수 있는 자판을 제작했을 때 시스템에서 지원을 하지 않는다면 정상적으로 문자가 출력되지 않는다. 다음 절에서는 소프트키보드 어플리케이션이 전달한 문자를 메시지 어플리케이션이 처리하는 과정을 설명한다.

2. 메세지 어플리케이션의 문자처리

메세지 전송 어플리케이션에서 문자를 화면에 표시하는 작업은 어플리케이션에서 담당하는데 기본적으로 Freetype 폰트 엔진을 사용한다. 그러나 Freetype 폰트 렌더러를 직접 수정하여 사용자화 하는 것은 시스템 접근의 위험 내포와 오랜 개발 시간을 필요로 하기 때문에 어플리케이션 수준에서 자체 키보드와 폰트 렌더러를 개발하는 방법으로 본 논문에서 제안한 시스템 수준의 힌디어 지원 방법을 검증하였다.

메세지 어플리케이션으로 전달된 문자열은 문자 코드로 표현되어 있는데, 화면에 힌디어를 표시하는 렌더링 과정에서 그립 인덱스로 변환된다. 그런데 힌디어는 자소조합에 따라 क(ka)+इ(i)=कि=किक(kai)와 같이 문자의 순서와 모양을 바꿔주어야 하는데, 앞장에서 설명한 바와 같이 그립 인덱스 변환은 테이블을 기준으로 실행되므로 그립 인덱스 변환테이블을 통해 이러한 특징을 쉽게 구현할 수 있다.

오토마타를 통해 유니코드를 그립 인덱스로 변환한 뒤 [그림 6]과 같이 시작 좌표와 그립 인덱스를 NDK만든 라이브러리로 보내면 출력 데이터를 받아와 화면에 문자를 출력하게 된다.

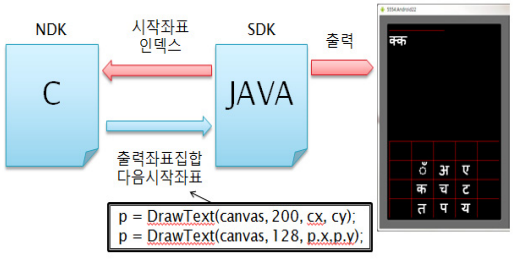


그림 6. NDK 와의 상호작용

어플리케이션 상에서 구현은 Freetype 라이브러리를 이용하여 외부 폰트를 불러오는 방법도 가능하나, 본 실험에서는 Windows Truetype[14] 힌디어 폰트를 사용하였다. 한편, Skia 라이브러리를 참조하면 Freetype 라이브러리의 기능들을 사용할 수 있는데, Freetype 라이브러리는 C/C++ 동적 라이브러리로 되어있기 때문에 Java 언어를 사용하는 안드로이드 SDK가 아닌 C언어를 사용하는 안드로이드 NDK를 사용하여 개발해야 한다. NDK의 설정까지 되었다면 Freetype 라이브러리로 폰트를 읽고 출력하고자 하는 문자의 그립 인덱스를 받아 출력한다. [그림 7]은 안드로이드 폰에서 구현된 힌디어 입력기로 문자를 입력하는 예를 보여준다. 화면의 밑줄 문자는 현재 입력되고 있는 힌디어 음절을 나타낸다.

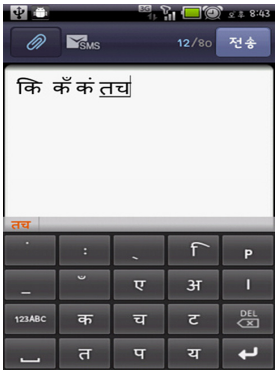


그림 7. 힌디어 입력 예

V. 결 론

문자를 처리하는 기능은 컴퓨터의 가장 기본적인 기능으로 기본적으로 안드로이드는 다국어를 지원하도록 설계되어 있다. 그러나 힌디어와 같은 일부 언어는 현재 안드로이드에서 완전하게 지원되지 않는다. 한글의 경우도 한글 고유의 특성 때문에 표준 안드로이드에서 지원되지 않고, 한글을 지원하기 위한 문자처리 라이브러리가 사용자화 되어있다.

본 논문에서는 힌디어를 시스템 수준에서 처리하기 위한 방법을 제안한다. 힌디어는 조합형 구조를 가지는 문자로 음절 인식을 위한 오토마타가 필요하며, 이를 안드로이드에서 구현하기 위해서는 문자-그립 매핑방법이 필요하다. 힌디어 폰트는 자음과 모음 각각에 대한 그립 이미지를 디자인하고 음절은 조합하는 방법을 사용한다. 자음과 모음에 대해 문자-그립 매핑은 매핑 테이블을 사용하고, 오토마타 상에서 매핑 시점을 제시하였다.

최근 세계적으로 스마트 폰이 급속히 보급되는 추세여서, 안드로이드 폰 시장도 빠른 속도로 커질 것으로 보인다. 한국은 안드로이드 폰을 생산하는 주요 나라 중의 하나이므로 안드로이드에서 다국어를 지원하는 기술을 확보하는 것이 세계시장으로 진출하는데 유리할 것으로 판단된다.

참 고 문 헌

[1] 고석훈, “안드로이드 플랫폼 동향”, 한국콘텐츠학회지, 제8권, 제2호, 2010.

[2] 김정훈, “구굴의 안드로이드와 안드로이드 마켓”, 한국콘텐츠학회지, 제7권, 제2권, 2009.

[3] 변정용, 유정원, “다국어 입력기에서 한글 입력의 최적화 방안”, 한국정보처리학회 추계학술발표회, 제12권, 제2호, 2005.

[4] 이고은, 이종우, “스마트폰 상에서의 웹 응용프로그램 개발환경 비교”, 한국콘텐츠학회논문지, 제10권, 제12호, pp.155-163, 2010.

[5] 이진영, 홍성용, 이시진, “임베디드 시스템에서 필획기반 다국어 입력 시스템”, 한국인터넷정보학회, 제8권, 제6호, 2007.

[6] 이진영, “모바일 기반의 다국어 입력 시스템”, 정보처리학회, 제15권, 제4호, pp.65-72, 2008.

[7] 서행정, *힌디발음 입문*, 한국외국어대학교 출판부, 2003

[8] 장충엽, *다국어 입력기*, IPC G06F 3/02, 공개번호 1020010003037, 2001.

[9] 정희성, *한글 및 다국어 문자 생성기*, IPC G06F 15/38, 11019940010886, 1994.

[10] 황경선(LG전자), *휴대전화 단말기 다국어 지원 시스템 및 방법*, IPC H04Q 7/24, 1020040059633, 2004.

[11] http://www.unicode.org/~core

[12] http://developer.android.com

[13] http://www.freetype.org

[14] http://www.microsoft.com/typography

저 자 소 개

김 재 혁(Jae-Hyeok Kim) 준회원

- 2011년 현재 : 공주대학교 컴퓨터공학과 재학

<관심분야> : 컴퓨터 그래픽스

맹 승 렬(Seung-Ryol Maeng) 정회원

- 2004년 2월 : KAIST 졸업(공학박사)
- 1994년 ~ 현재 : 공주대학교 컴퓨터공학부 교수

<관심분야> : 컴퓨터 그래픽스, 계산기하학