무선 단말기 상에서의 효율적인 이미지 및 폰트 처리

강신상[O] 옥경달 이상범
단국대학교 전자계산학과
{kashsa[O], okdal, sblee}@dankook.ac.kr

Manipulation of Image and Font for Mobile Stations

Shinsang Kang[0], Kyoungdal Ok, Sangbum lee
Dept. of Computer science, Dankook University

요 약

본 논문에서는 무선 단말기에서 이미지 및 폰트를 효율적으로 처리할 수 있는 비트맵 처리 기법을 제안하고 이를 이용한 이미지 및 폰트 처리 시스템을 소개 하고자 한다. 초기에는 컴퓨터상에 문자를 표현하기 위하여 아스키 코드가 개발 되었고 이에 표현하지 못하는 부분을 더하고자 유니코드가 개발 되어 사용하고 있으나 코드에 대한 표준화 작업은 지금도 계속 되고 있다. 이러한 아스키 코드와 유니코드를 무선 단말기상에 적용하고자 하는 시도는 많이 했지만 시간과 노력이 많이 요구되는 비효율 적인 작업이 계속 되어 온 것도 사실이다. 본 논문에서는 이러한 문제점을 해결하고자 아스키 코드 및 유니코드를 무선 단말기에 적용시키는 일련의 과정을 단축하고 능률적인 처리시스템을 소개하고자 한다. 본 연구에서는 이미지 및 폰트 처리의 불필요한 작업을 단축하여 최소 비용의 처리 시스템을 설계 및 구현하였다.

1. 서론

1996년부터 무선 단말기 보급이 늘어나기 시작하면서 국내에서만 가입자가 3천만이 넘어섰다. 국내 무선단말기 제조 업체는 국내는 물론 해외 시장까지 점유율을 높이고 있는 현 시점으로 미루어 볼 때 무선 단말기에서의 한글뿐만이 아니라 여러 국가의 다양한 형태의 언어를 처리함과 동시에 각기 다른 소비자들의 요구를 만족시키기 위한 사용자 인터페이스를 필요로 하고 있다.

이러한 문자와 인터페이스 작업에는 상당히 많은 폰트와 이미지 처리를 요하게 된다. 이는 우선 폰트 처리에 있어서 상당한 작업과 노력을 요구하게 되는데 한 글자(즉, 8비트 아스키코드로 표현 할 수 있는 문자와 2바이트 유니코드로 표현 될 수 있는 한 문자)를 처리하는 시간이 대략 3분에서 5분까지 걸리고 있다. 한자를 예를 들면 무선 단말기상에 적용되는 한자의 글자수는 대략 3만자 가까이 됨을 생각해보면 결코 적게 걸리는 시간이라 말 할 수 없을 뿐더러 수시로 바뀌는 것을 감안하면 이 작업에 필요로 하는 시간과 비용은 상당할 것이라 예상된다.

또한 한자 뿐만이 아닌 세계 여러 나라의 언어를 모두 무선 단말기에 적용을 시키고자 한다면 이에 필요한 작업 시간 엄청나게 늘어나게 되어 이러한 폰트 처리에만 전적으로 담당하는 업체가 생길 정도가 되었다. 이미지 또한 마찬가지로 수시로 업데이트 해야만 하기 때문에 많이 시간이 소모되고 있다.

본 논문에서는 기존의 시스템을 분석하여 불필요한 작업을 없애고 작업의 능률과 효율을 극대화 시키기 위하여 무선 단말기 개발 과정의 중요한 과정인 이미지 및 폰트를 처리할 수 있는 시스템을 소개하고자 한다. 이 시스템은 다양한 언어를 일괄적으로 한꺼번에 단말기에 적용 할 수 있는 데이터로 변환을 할 수 있고 비트맵으로는 단말기에서 사용되는 데이터로 변환이 되지 않던 비트맵 폰트를 적용 함으로써 한층 향상된 시스템 임을 입증 할 수 있을 것 이다.

현재 무선 단말기 제조회사는 위의 전 과정을 상당한 시간과 인력을 들여가며 무선 단말기에 적용을 하고 있으며 그에 따라 상당한 비용이 지출 되고 있다. 따라서 본 시스템은 단말기 제조회사의 경비와 인건비를 줄이는 결과를 가져올 수가 있다.

우선 본 논문은 2장에서는 관련 연구인 아스키 코드와 유니코드 부분에 대해서 소개한다. 3장의 구현 부분에서는 설계된 시스템에 대하여 소개하고 또한 시스템에 대한 장단점을 설명 할 것 이다. 마지막으로 4장에서는 향후 과제 및 결론으로 본 논문을 마치고자 한다.

2. 관련 연구

◆ 아스키(ASCII)코드

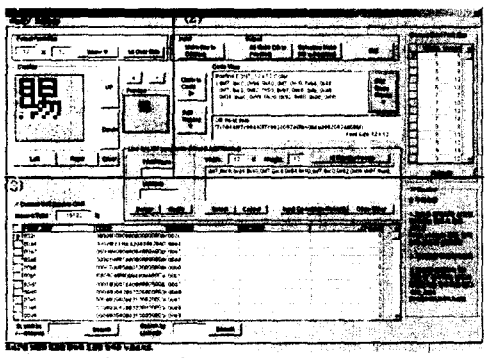
아스키 코드(ASCII : American Standard Code for Information Interchange)는 7 개의 비트의 조합으로 128 개의 문자를 표시할 수 있는 코드로 미국 규격

협회인 ANSI(American National Standard Institute)에 의해 개발되었다.
데이터 표현은 7 비트 그리고 오류 검사용 패리티 비트를 하나 추가하여 1 바이트로 표현된다. 영문자나 기호는 그 개수가 비교적 많지 않으므로 7 비트만으로도 0 과 1 의 조합들을 각각 부여할 수 있지만, 한글의 경우 조합 가능한 문자의 개수가 일 만개 이상이므로 아스키 코드로도 모두 표현할 수 없다. 따라서 아스키 코드가 아닌 다른 코드들이 나오게 되었다. [1][2]

◆ 유니코드(unicode)
한글, 한문, 일본어는 2바이트로 문자를 표현해야 하기 때문에 아스키 코드로는 표현할 수 없다. 이를 해결하기 위한 방안이 유니코드이다. 미국의 대표적인 소프트웨어 회사들이 세계의 모든 문자를 하나의 부호로 지원 할 수 있도록 만든 산업계 표준 코드이다. 65,536(2^{16})가지의 서로 다른 문자를 표현할 수 있다. [3]

3. 시스템의 설계 및 구현

본 시스템은 Visual C++를 이용하여 구현하였으며 윈도우기반의 PC에서 실행된다. 시스템의 전체적인 인터페이스는 [그림1]와 같으며 좌측상단 (1) 부분은 폰트 입력을 우측상단 (2) 부분은 데이터 출력을 그림 하단 (3) 부분은 데이터 저장부분을 표현한다. 이에 대한 자세한 설명은 아래에 계속된다.



[그림 1] 폰트 시스템 구성도

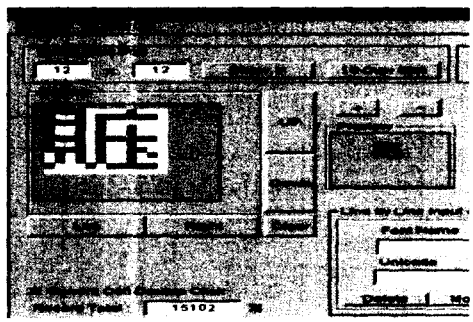
아래 [그림2]는 비트맵 폰트에 비트맵 폰트에 찍히는 데이터 값을 표현 한 것이다. 그림에 보이는 데이터 값을 적용하면 해당 부분에 폰트가 찍히게 된다 두 점이나 그 이상의 부분에 폰트를 찍으려 한다면 해당 데이터 값을 더해주기만 하면 된다.
예를 들어 만약 8비트 모두 폰트를 찍는다고 한다면 데이터 값은 Oxff 가 되는 것이다. [4]
이렇게 실제로 시스템에 적용한 것이 [그림 3]이다.
폰트 시스템에서 이와 같이 폰트를 찍어주는 역할을 하는 부분이 [그림 1]의 전체 부분에서 (1)부분에 해당된다.

Oxff

Ox80	Ox40	Ox20	Ox10	Oxf8	Oxf4	Oxf2	Oxf1

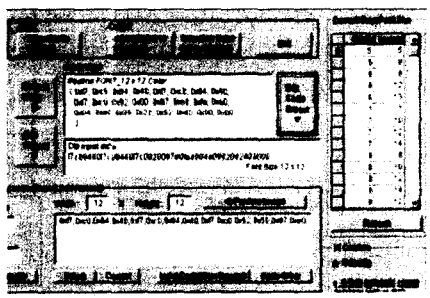
[그림 2] 8비트 데이터의 구성

[그림 3]에서 한자가 쓰여있는 곳이 폰트 입력을 하는 부분으로서 화면에는 12 x 12기준으로 한 글자를 입력한 모습이다. 글자입력은 마우스로 클릭 및 드레그로 문자를 입력하도록 구성하였다.
오른쪽에는 미리 보기 창을 만들어 실제로 입력하는 것과 보여 지는 것에 대한 차이를 미연에 방지토록 하였다



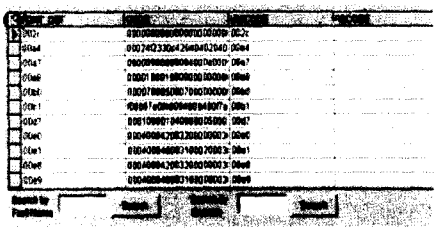
[그림 3] 폰트 입력

[그림 4]는 [그림 3]에서 입력한 글자를 16진수 데이터 값으로 출력한 모습이다. 출력형태는 실제로 무선 단말기에 입력되는 #define 선언문까지 출력 되도록 한 것과 순수 데이터 값만 출력되도록 한 것 두 가지의 형태로 출력되도록 하였다.



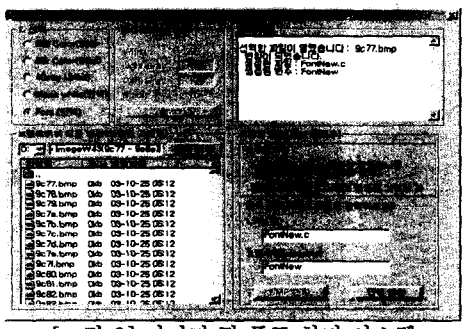
[그림 4] 데이터 출력

이 시스템은 데이터의 형태와 모습 그리고 유니코드 값까지도 모두 데이터베이스에 저장되도록 하였다.
[그림 5]는 그렇게 저장되어있는 데이터들의 모습을 보여주도록 한 것 이다. 데이터베이스는 MS Office에서 기본적으로 제공 되어지는 Access를 사용하였고 데이터 베이스와의 연동은 ODBC를 사용하여 연결하였다.



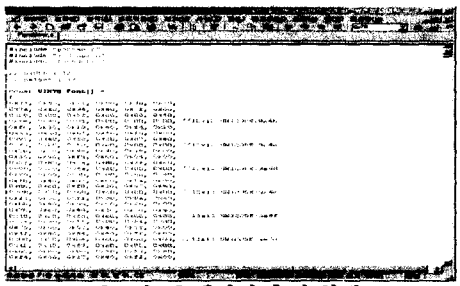
[그림 5] 데이터 저장부분

유니코드는 ISO/IEC 10646-1에 의거 하여 자동적으로 해당 코드가 저장되도록 구성 하였고 유니코드 표현 방법으로는 UCS(Universal Multiple-Octet Coded Character Set) 을 이용하였다.
이미지 및 폰트 처리 시스템은 BMP파일을 읽어 들여 16진수 데이터 값으로 변환 해주는 역할을 하는 시스템[그림 6]으로 다양한 형태의 데이터 변환을 할 수 있다.



[그림 6] 이미지 및 폰트 처리 시스템

이 시스템에는 256 및 65k color bit map, 4Gray, black & white(단색), 폰트 등의 비트맵을 처리 할 수 있도록 구성하였으며 처리 량은 컴퓨터의 사양에 따라 달라지겠지만 대략 PIII 700 MHz의 사양을 가진 컴퓨터에서 30000 Character를 처리하는데 2분이 채 걸리지 않았다. 출력 형태는 그림 7과 같이 하나의 C파일의 형태로 출력되도록 하였으며 각각의 데이터는 정렬되어 파일내부에 저장되도록 하였다.



[그림 7] 데이터 출력 형태

[그림 7]과 같이 출력되도록 한 것은 무선 단말기 상에서의 데이터 저장 형태가 [그림 7]의 출력 형태를 갖기 때문에 출력 형태를 이와 같이 하였다.
그렇게 함으로써 출력된 그대로를 곧바로 무선 단말기에 적용하면 되는 것이다.

4. 향후 과제 및 결론

본 논문에서 구현된 시스템은 유니코드에 정의된 character중 무선 단말기에 적용 시킬 수 없거나 쉽게 표현되지 않는 문자들에 대한 문제점을 해결하는데 주안점을 두고 구현 되었으며 또 하나는 저비용 저 시간으로 많은 데이터를 처리 할 수 있도록 하는데 주안점을 두고 시스템을 구현하였다.
이러한 시스템을 개발 사용해봄으로써 많은 시간과 비용을 단축 시키는데 일조를 한 것은 사실이나 아직까지는 BMP 파일을 만드는 것 또한 상당한 비용과 인력을 필요로 하고 있다. 향후 과제로써 앞으로는 이 BMP 파일을 생성하는 것에도 비용과 인력을 단축하여 단기간에 생성 할 수 있는 시스템이 필요하며 본 논문에 소개된 시스템에 대한 통합과정도 앞으로 이루어져야 할 과제이다.
본 시스템 중 폰트 시스템은 데이터베이스에 데이터가 저장되어있으므로 유지보수 측면에 대한 유동적인 대처가 가능하지만 이미지 및 폰트 처리 시스템은 그렇지가 못한 것이 사실이다. 그러므로 유지 보수 측면의 기능적인 보수 또한 향후 과제로 남아있다.

참고 문헌

[1] ISO/IEC-10646 Universal Multiple-Octet Coded Character Set (UCS)에 대해서, 정주원, 1995년 11월 11일

[2] ISO/IEC 10646-1 " Information technology - Universal Multiple-Octet Coded Character Set (UCS) - " , First edition pp.1-15, May. 1993

[3] " 컴퓨터 속의 한글이야기" 김 경 석 ch.21 전북대학교 1995.03. 영진출판사

[4] " C로 구현한 OUTLINE FONT EDITOR" 백성수 著 April, 1994 높이깊이 출판사.