한글 X 윈도우 시스템 및 그래픽 사용자 접속에 관한 연구

조명진○, 하종성*, 김판건*, 전길남*
○ (주)삼보컴퓨터, * 한국 과학 기술원

Study about Korean X Window System and Graphical User Interface

Myungjin Cho○, Jongsung Ha*, Pankeon Kim*, Kilnam Chon*
○ TriGem Inc., *KAIST

요 약

본 논문에서는 한글 X 윈도우 시스템의 개발 과정을 소개하고 이를 바탕으로 한 GUI의 한글화 방안을 제시한다. X 윈도우 시스템은 이식성이 높고 다양한 사용자 접속을 제공한다. 특히 소스가 거의 공개되어 있어 UNIX 워크스테이션의 표준 윈도우 시스템으로서의 위치를 굳혀가고 있다. 한글 X는 여기에 한글의 입출력 기능을 추가한 것이다. 현재 UNIX와 X 윈도우를 기반으로 한 그래픽 사용자 접속(GUI)을 표준화하려는 움직임이 세계적으로 높아가고 있다. OSF와 UI의 두 단체에서 각각 Motif와 OpenLook을 표준 GUI로 내놓고 있는데 이 두 GUI는 모두 나름대로의 지지 기반을 가지고 사용될 것으로 보인다. 때문에 한글 GUI의 개발은 시급하고 중요한 문제라고 할 수 있다. 한글 X의 개발은 X 윈도우 환경에서 한글을 사용할 수 있게 되었다는 것뿐만 아니라 GUI 한글화의 기반을 마련했다는 점에서 그 의의를 갖는다.

I. 개요

X 윈도우 시스템은 MIT의 Athena 프로젝트와 Digital Equipment Corporation(DEC), 그리고 다른 여러 소프트웨어 회사들의 합작에 의해서 개발되었다. MIT의 Robert Scheifler와 Jim Gettys가 주축이 되어 개발한 것으로 이식성이 높고 Xlib나 툴키트의 다양한 사용자 접속을 제공한다.

Multi-user 및 Multi-tasking 운영체제로서의 UNIX가 각광을 받고 있고, 이를 기반으로 하는 Workstation의 효용성이 크게 대두되고 있는 가운데에 이러한 Workstation의 표준 윈도우 시스템으로 받아들여지고 있는 X 윈도우 시스템의 한글화는 한국적 Workstation의 개발에 있어서 중요한 의미를 갖는다.

한글 X 윈도우 시스템[16]은 X 윈도우상에서 한글의 처리가 가능하도록 한 것으로서 한글 비트맵 폰트와 이를 출력할 수 있는 라이브러리 및 여러 윈도우나 응용 프로그램에서 공동으로 사용할 수 있는 한글 자동 생성기를 제공하므로 X 환경위에서 한글 응용 툴들을 개발할 수 있는 기반이 된다. 그리고 대표적인 응용 툴인 한글 터미널 에뮬레이터(*hterm*)를 개발하여 윈도우내에서 한글 UNIX의 명령어들을 처리해 줄 수 있도록 한다.

현재 UNIX와 X를 기반으로 한 워크스테이션에서의 그래픽 사용자 접속(Graphical User Interface: GUI)에 대한 표준화의 움직임이 국제적으로 활발해지고 있는데 그러한 추세속에서 Motif[2]나 OpenLook[3]이 양대 표준으로 두각을 나타내고 있다. Motif는 UNIX 워크스테이션의 표준 윈도우 시스템인 X위에서 만들어졌고, OpenLook 역시 X 툴키트를 대부분 사용해서 만들어져 있다. 그런데 Motif나 OpenLook은 모두 영어 문화권에 맞게 만들어져 있기 때문에 표준 모형에 맞는 한글 그래픽 사용자 접속을 개발하여야만 한다.

본 논문의 2장에서는 X 윈도우 시스템의 간략한 소개를 하고, 3장에서 한글 X를 개발한 과정을 기술하였다. 4장에서는 GUI 표준화의 추세를 살펴보고, 이에 대한 한글화의 중요성 및 한글화 방안에 대해서 5장에서

기술하였다. 마지막으로 6장에서는 이와같은 한글화 작업의 의의와 문제점들, 그리고 비슷한 문자 체제를 가지고 있는 중국이나 일본등과의 협력 작업에 대한 가능성등을 살펴보았다.

II. X 윈도우 시스템의 특징 및 구조

X 윈도우 시스템은 다음과 같은 장점들로 인하여 현재 UNIX에서 사용할 수 있는 여러 윈도우 시스템중에서 거의 표준으로 그 위치를 굳혀가고 있다.

○ client-server 모형 : 모든 디스플레이 장치는 서버(server)를 갖는다. 서버는 많은 고객(client)으로부터의 요구를 다중화(multiplex)하고, 키보드나 마우스 입력을 통합(demultiplex)하여 사용자의 입력을 적당한 고객에게 전달해 주는 역할을 하고, 고객은 X 환경에서 돌아가는 여러 응용 툴들을 말한다.

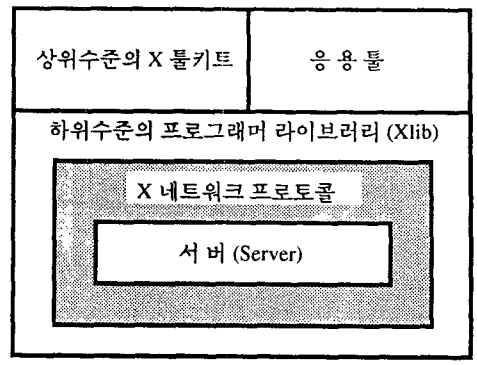
○ 높은 이식성 : 하드웨어에 의존적인 부분은 서버에 모두 있으며, 고객 부분은 디바이스에 대하여 완전한 독립성을 갖고 있다. 서버 부분도 디바이스에 의존적인 부분을 최소한으로 줄여 놓아서 어떤 기계에라도 이식하기에 쉽다.

○ Network-transparency : 어떤 기계에 운영되고 있는 응용 프로그램이 다른 기계에 있는 디스플레이 장치를 이용할 수 있으며, 이때 이 두개의 하드웨어는 똑같은 구조나 OS를 가질 필요가 없다.

○ 다양한 사용자 접속 : X 환경위에서는 정형화된 한 종류의 사용자 접속만이 아니고 다양한 사용자 접속이 가능하다. X 라이브러리(Xlib)나 X 툴키트와 같은 안정된 라이브러리를 갖고 있기 때문에 이를 이용하여 다양한 사용자 접속을 만들어낼 수 있다. 그밖에도 높은 확장성을 갖고 있으며 고도의 2차원 그래픽을 제공한다.

X는 개발 과정에 따라 크게 두 부분으로 나누어지는데 *core* 부분과 *contribution* 부분이 바로 그것이다. *core* 부분은 애초에 MIT와 DEC에 의해서 개발된 X의 중심적인 부분이고, *contribution* 부분은 그 이후에 X의 개발에 참여했던 회사들이 만든 응용 툴과 다른 회사나 연구 개발 단체들이 자신들의 필요성에 의해서 X 환경위에 개발한 여러 응용 툴들이 있는 부분이다.

X는 그림 2-1에서 볼 수 있듯이 응용 소프트웨어의 사용자와 프로그래머에게 아주 편리한 환경을 제공해 준다.



< 그림 1-1 > X 윈도우 시스템의 구조

가장 낮은 단계에 X의 서버가 깔려 있고, 그 위에 네트워크 프로토콜이 자리잡고 있다. 네트워크 프로토콜은 X의 서버와의 유일한 인터페이스이다.

대부분의 X 응용 툴들은 프로그래밍 인터페이스인 Xlib와 X 툴키트를 이용한다. Xlib와 X 툴키트를 통해서 응용 프로그램들은 네트워크 프로토콜과의 인터페이스를 이루기 때문에 프로그래밍 작업이 수월해진다. 이러한 X 툴키트 인터페이스는 기본적으로 X의 응용 프로그래머들이 그들의 상상력과 취향에 따라서 다양한 모양의 사용자 접속을 만들 수 있는 환경을 제공해 준다. 어떤 한 형태로 규정되어있는 사용자 접속만을 제공하는 것이 아니고 여러가지 다양한 형태의 사용자 접속을 제공하는 것이다. 이는 X의 기본 철학이기도 하다.(The central dogma of X is that the base window system provides *mechanism, not policy*)[5]

III 한글 X 윈도우 시스템

X 윈도우 시스템의 한글화 과정은 대략적으로 3 단계의 작업을 거쳤다. 먼저 X에서 사용할 수 있는 형태의 한글 폰트를 만들고 이를 X의 버전 11 이상에 있는 16 비트 코드 처리 루틴을 이용해서 출력할 수 있는 기능을 추가하였다. 입력시에는 한글에 대한 event 처리의 기능과 함께 한글 자동생성기(automata)를 만들고, 마지막으로 X상의 한 응용 툴로서 한글 단말기 에뮬레이터(hterm)를 개발하는 과정을 거쳤다.

1. 한글 폰트 및 화면 처리

X에서는 폰트를 소프트웨어적으로 관리하고 있다.

폰트의 크기나 형태별로 각각의 화일로 만들어서 가지고 있으며 지정된 폰트의 코드 값을 갖고서 폰트 비트맵을 읽어서 화면에 디스플레이하거나 화일에 출력하는 방식을 쓰고 있다.

한글의 경우에는 폰트 화일의 크기가 영문 폰트 화일과 비교가 되지 않을 정도로 크기 때문에 보통 ROM안에 데이타로 넣어 놓고 지정된 한글 코드에 해당하는 폰트를 가져오는 방법을 쓰고 있다. 그러나 X에서는 여러개의 윈도우에서 각각 폰트를 다루기에 편리하다는 점과 하드웨어에 의존적인 부분을 줄인다는 점에서 소프트웨어적인 방법을 쓰고 있다. 따라서 한글의 경우도 X의 영문 폰트의 경우와 마찬가지로 특정한 형태의 자료 구조를 갖는 폰트 화일로 만들어야 한다.

X에서 폰트를 만들기 위해서는 Adobe Inc.의 Bitmap Distributed Format(BDF) 2.1[7]을 사용한다. BDF 2.1이란 Adobe System Inc.의 character bitmap distribution format version 2.1을 말한다. 이 형식은 인간과 컴퓨터 모두에게 이해하기 쉽도록 만들어졌다. BDF에 맞는 형태로 만들어진 폰트는 X에서 제공하는 폰트 컴파일러 *bdftosnf*를 사용해서 Server Natural Format(SNF)[8]의 형태로 바꾼다. 폰트 화일이 SNF의 형태로 만들어지면 지정된 디렉토리에 등록하여 X에서 사용할 수 있다.

본 한글화 작업에서는 KS-5601 완성형 코드 체계[9]에 맞는 순서로 2350자의 한글과 상용한자 4888자, 그리고 한글 자모등을 비롯한 특수 문자 986자를 모두 포함하는 BDF 화일을 만들기 위하여 기존의 다른 형태의 폰트 화일을 변환하는 프로그램을 개발하였다. 따라서 이밖의 폰트를 X에서 사용하려면 BDF로의 변환 프로그램을 먼저 개발하여야 한다.

이렇게 개발된 한글 폰트는 프로그래머가 사용할 수 있는 라이브러리 인터페이스를 통하여 화면에 출력시킬 수 있어야 한다. X의 버전 11 이상에서는 1 바이트로 처리가 가능한 알파벳 문화권의 언어들과는 달리 2 바이트 이상을 사용해서 구현할 수 있는 언어들, 예컨대 한국어나 중국어, 일본어등과 같은 동양권의 언어를 지원하기 위한 라이브러리들이 개발되어 있다.

이러한 라이브러리들은 모두 기존의 8 비트를 지원하던 라이브러리들을 수정하고 루틴 이름의 끝에 "16"을 붙여서 16 비트를 처리한다는 의미를 나타내고 있다. 예컨대 8 비트로 된 코드를 가지고 가서 그에 해당하는 글자를 화면에 디스플레이하는 기능을 가진 **XDrawString**과 같은 경우에 16 비트 코드에 대해서 똑 같은 기능을 수행하는 **XDrawString16**이라는 라이브러리가 있다.

X에서 이용할 수 있는 16 비트 처리 루틴들은 다음과 같다.

XQueryTextExtents16() : round trip의 방식으로 16 비트 문자나 문자열의 크기에 대한 정보를 준다.

XTextExtents16() : round trip의 방식을 사용하지 않고 16 비트 문자나 문자열의 크기에 대한 정보를 준다.

XTextWidth16() : 16 비트 문자의 넓이에 대한 정보를 준다.

XDrawImageString16() : 16 비트 문자와 배경을 화면에 출력한다.

XDrawString16() : 16 비트 문자를 화면에 출력하는데 배경은 출력하지 않는다.

XDrawText16() : 16 비트의 텍스트를 화면에 출력한다.

2. 한글의 자동 생성

2.1 한글 입력의 특성

한글은 표음 문자이면서도 모아쓰기를 하기 때문에 알파벳 문화권의 다른 문자들과는 달리 한 번 이상의 타자에 의해서 한 음절이 완성된다. 이 때 글자가 완성되는 과정을 스크린위에 보여 주어야 하고, 잘못 타자한 경우에 Backspace 키를 쳤을 때 영문과는 달리 화면이나 데이타 화일에서 2 바이트 만큼 뒤로 가야하는 등의 특수성을 가지고 있다. 보통 한글을 지원하는 컴퓨터나 단말기들은 이러한 기능을 하드웨어 ROM에 내장하여 처리하고 있기 때문에 한글 응용 툴에서는 완성된 음절의 한글 코드만 처리할 수 있도록 프로그래밍을 하면 된다.

그러나 X와 같은 Multi-Window 시스템에서는 여러 개의 윈도우에서 동시에 한글 작업을 하는 경우가 있을 수 있으므로, 각 윈도우마다 한글 음절의 완성 과정을 보여주어야 함은 물론이고, 한 윈도우에서 한글을 입력하던 도중에 다른 윈도우에 가서 작업을 하게 될 때에는 그 때까지의 한글 입력 상태를 가지고 있어야만 한다. 이러한 점들을 생각해 볼 때 하드웨어로 한글의 입력을 지원한다면 대단히 복잡한 ROM 프로그래밍과, 윈도우 시스템을 사용하지 않을 때의 낭비를 감수해야만 할 것이다.

2.2 한글 자동 생성기

X에서 각 윈도우는 자신의 상태를 나타내는 자료

구조체를 갖고서 그것을 이용하여 Multi-Window를 구현하고 있다. 이 같은 자료 구조체에 한글에 대한 정보를 넣어서 처리한다면 하드웨어로 처리하는 것보다는 프로그래머에게 융통성있는 방법이 될 것이다. 따라서 윈도우내에서의 한글입력상태에 대한 정보를 갖고 있는 자료 구조체와 한글 자동 생성기의 인터페이스인 라이브러리를 이용하여 한글 입력이 가능하도록 했다.

한글 자동 생성기는 한글 자모를 입력받아서 그때 그때의 상태와 다음의 행동을 결정해 주는 것을 말한다. 한글 자동 생성기에서는 먼저 한글의 입력 상태에 대한 정보를 갖고 있는 자료 구조체 Korean_handle을 다음과 같이 설정하였고 이를 윈도우와 자동 생성기간에 인수로서 주고 받으며 한글의 입력상태에 대한 정보를 유지하고 화면에 echo, 또는 local echo를 한다.

```
struct korean_handle {
  int mode:    /* 한글, 영문 모드 */
  int state:   /* 상태변환 테이블에서 얻은 입력상태 */
  char buffer[6];
               /* Ascii 코드로 저장된 한글자모 */

  unsigned char echo[6];
               /* 결정된 음절의 한글코드 */

  unsigned char local_echo[6];
               /* 화면처리용의 한글코드 */
  }
```

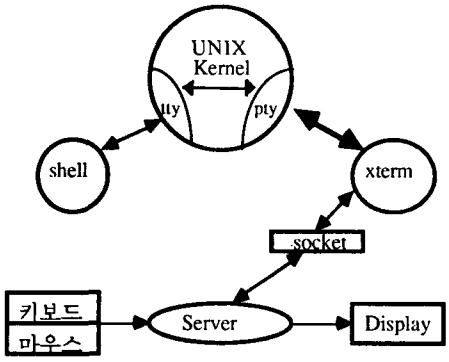
한글 자동 생성기를 사용하기 위하여 다음과 같이 라이브러리의 형태로 한글 자동 생성기를 만들었기 때문에 X상에서 새로운 응용 툴을 만들고자 할 때에는 이 루틴들을 적절하게 호출하여 사용할 수가 있다.

```
void auto_init(*handle)
     struct korean_handle handle;

void auto_get(c, *handle)
     unsigned char c;
     struct korean_handle handle;

void auto_flush(*handle)
     struct korean_handle handle;
```

이같이 각 루틴들은 korean_handle이라는 자료 구조체를 인수로 사용하고 있으며 이 자료 구조체는 각 윈도우안에 보관되어 있다. auto_init()는 한글 자동 생성기를 사용하기 위해 맨 처음에 호출하는 루틴으로 한글 입력 정보를 가지고 있는 자료 구조체를 초기화한다. auto_get()은 입력받는 한글 자소에 대해서 그것을 인수로 해서 한글 음절의 완성 과정을 처리하며, auto_flush()는 한글 자동 생성기에서 빠져나오기 전에 결정되지 않은 상태로 버퍼에 남아있는 자료들을 처리하고 한글 자동 생성기의 상태를 초기화한다.

3. 한글 단말기 에뮬레이터

한글 단말기 에뮬레이터(*hterm*) 는 X 윈도우 시스템에서 한글 처리를 가능하게 해 주는 단말기 에뮬레이터이다. X에는 본래 *xterm*이라는 영문 단말기 에뮬레이터가 있어서 Tektronix의 4015 터미널과 DEC의 VT102 터미널을 에뮬레이션해 주는데 *hterm*은 이 *xterm*에 한글을 처리할 수 있는 기능을 추가하여 만들었다.



< 그림 3-1 > *xterm*의 동작도

*xterm*은 그림 3-1과 같이 *shell*을 가동시키고 윈도우내에서의 모든 입력을 가상 터미널 구동기(pty)를 통해서 *shell*에게 인계한다. *shell*에서 처리된 결과는 다시 가상 터미널 구동기를 거쳐서 *xterm*이 되돌려 받고 이를 서버에게 전달해 줌으로써 서버가 처리 결과를 화면에 출력하게 된다.

*xterm*에서 문자나 문자열의 출력은 *charproc.c*에 코딩되어 있다. 여기에 한글 폰트를 설정해주고 데이타의 첫번째 비트를 검사해서 한글 여부를 알아내서 한글일 경우 16비트 코드 처리 루틴을 호출해서 화면에 출력할 수 있는 기능을 추가했다.

입력 기능은 *input.c*에서 한글 자료 구조체 *korean_handle*을 포함함으로써 *xterm*이 한글 입력 상태를 유지시키고 한글 입력의 시작과 끝을 인식할 수 있는 한글 event 처리 루틴을 만들어서 한글 입력시에는 한글 자동 생성기 라이브러리를 호출하여 처리할 수 있도록 하였다.

그밖에도 한글 UNIX의 한글 *shell*을 가동시키도록 하고 폰트의 크기에 따른 커서의 크기나 입력시의 커서의 이동도 한글의 특성에 맞게 처리해 주었다.

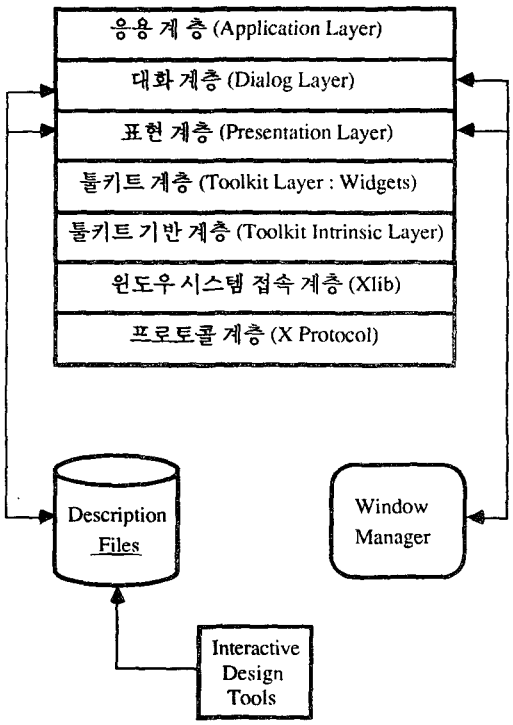
IV. 그래픽 사용자 접속의 국제 경향

1. 7 계층의 참조 모형

사용자 접속이란 컴퓨터와 사용자간의 공유 부분으로서 키보드나 마우스를 이용해서 화면에 어떤 결과를 출력하기 위해 하게 되는 입력을 말하거나 응용 툴 프로그래머가 새로운 툴을 개발할 때 사용하게 되는 부분을 말한다. 이 부분을 흔히들 응용 프로그래머 접속(Application Program Interface: API)이라고도 한다.

GUI는 사용자 접속중에서도 사용자가 화면을 보면서 입출력을 하게 되는 부분으로 윈도우나 스크롤바, 메뉴의 모양이나 이를 만들어내기 위한 도구로서의 툴키트등을 말하는 것이다.

현재 UNIX의 표준화에 대한 세계적 동향이 이제는 OS의 범주를 넘어서서 그래픽 사용자 접속(Graphical User Interface: GUI)에 관한 부분으로 넘어가고 있다. 이와 관련해 National Institute of Science and Technology(NIST)에서 제시한 사용자 접속 참조 7 Layer 모형[10]을 X 윈도우 시스템과 관련을 시켜서 간단히 살펴볼 수 있다.



< 그림 4-1 > 사용자 접속 참조 모형

먼저 그림 4-1에는 나타나있지 않지만 가장 하위 수준에는 물론 컴퓨터 하드웨어가 있고, 그 위에 운영체제와 X 윈도우 시스템의 서버가 순서대로 자리잡고 있다. 그 위에 X의 서버와 X 응용 툴간의 상호 통신이 가능하게 해주는 X 프로토콜이 있고, 이 프로토콜을 이용해서 X의 서버에게 어떤 종류의 동작을 요구할 수 있는 하위 수준의 접속 부분인 Xlib가 있다.

X에서는 응용 툴의 개발에 좀 더 편리한 환경을 제공하기 위해서 툴키트 계층을 제공하고 있는데 이를 위한 툴키트 기반과 툴키트의 계층을 갖고 있다. 툴키트 기반 계층은 사용자 접속의 형태를 정의하기 위한 일반적인 틀을 가지고 있으며, 이의 상위 계층에 구체적인 사용 가능 형태로의 사용자 접속 틀을 가지고 있는 툴키트 계층이 자리잡고 있다.

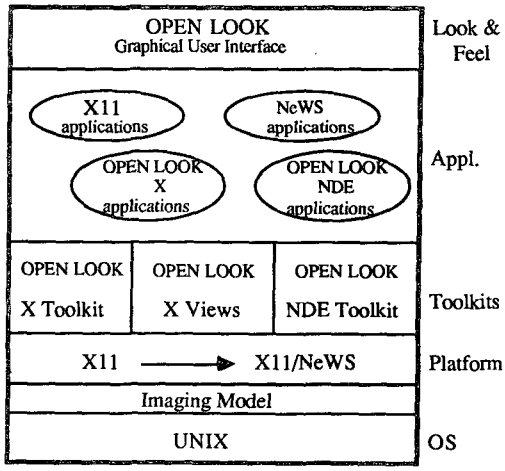
프레젠테이션(Presentation) 계층은 이 같은 툴키트 계층을 이용해서 여러 응용 툴에서 요구될 수 있는 다양한 형태의 사용자 접속 모형들을 갖고 있다.

그리고 사용자에게 컴퓨터와의 좀 더 편리한 접촉을 가능하게 해 줄 수 있는 기능을 위한 대화 계층(Dialog)이 있어 대화 형식의 동기화 처리를 다룬다. 이러한 여러 계층의 위에 비로소 최종 사용자의 요구를 충족시켜줄 수 있는 응용 계층이 자리잡고 있다.

2. Motif 및 OpenLook

Open Software Foundation(OSF)과 Unix International(UI)에서는 그림 4-1의 모형을 기반으로 한 Motif와 OpenLook을 그들의 GUI 표준으로 제시하고 있다.

OSF는 IBM과 HP등이 중심이 되어 결성한 단체로 X 윈도우를 기반으로 하여 개발한 Motif를 표준 그래픽 사용자 접속으로 내놓고 있고, UI 재열에서는 AT&T의 지원을 받아서 개발한 OpenLook을 내놓고 있다.



< 그림 4-2 > OpenLook의 구조[14]

Motif는 IBM의 Presentation Manager(PM)와 외형적인 모습을 유사하게 만들었으며 X의 툴키트를 지원한다. OpenLook은 그림 4-2에서 볼 수 있듯이 3 종류의 툴키트를 사용해서 만들었는데 이 중 X 툴키트와 X View 툴키트가 X를 기반으로 하고있다. 현재까지는 Motif가 OpenLook보다는 더 많이 보급되어 있는데 앞으로 양자중의 하나가 표준으로 결정되기보다는 두 가지 다 나름대로의 사용기반을 갖게 될 것으로 보인다.

Motif의 툴키트와 그림 4-2에서 나타난 OpenLook의 툴키트는 앞에서 제시한 7 계층 모형의 툴키트 계층에 속한다고 할 수 있고 OpenLook의 FORM widget과 Motif의 사용자 접속 언어(User Interface Language: UIL)는 프레젠테이션 계층에 속한다. UIL은 상위 레벨의 응용 툴과는 관계없이 사용자 접속을 만들 수 있는 언어이다.

현재 세계적인 추세로는 이러한 상위레벨 GUI에 대한 표준화 작업, 다시 말해서 어떤 UNIX, 어떤 윈도우 시스템, 어떤 응용 툴을 사용하느냐에 관계없이 같은 형태의 사용자 접속을 갖게되는 사양에 대한 표준을 만들려고 하는 연구가 일고 있다. 처음에는 UNIX에 대한 표준화의 동향으로 시작되었는데 지금은 UNIX상에서의 윈도우 시스템과 그보다 상위 레벨의 사용자 접속에까지 범위를 넓혀가고 있다.

특히 OSF에서는 Motif의 사양에 세계 각국의 언어에 대한 표준화 사양을 추가하려는 움직임이 있다. 이에 대해서는 다음 장에 좀 더 자세히 기술하겠지만 한글화 작업시 참조해야 할 일이라 할 수 있다.

V. 한글 그래픽 사용자 접속

1. 중요성

앞장에서 살펴보았듯이 OSF와 UI에서 제시한 두 GUI 모두 X 윈도우 시스템을 그 기반으로 하고 있을 만큼 X는 이미 UNIX위에서의 표준 윈도우 시스템으로 굳어졌다고 할 수 있다. GUI의 표준화에 있어서는 Motif가 OpenLook보다 조금 우세한 위치에 있는데 한글화 작업도 이같은 세계적 추세에 발맞추어서 나아가야 할 필요성이 있다.

특히 OSF 계열에서는 운영체제와 GUI의 국제적 표준화를 위해 현재 영어 문화권에 맞게 만들어져 있는 여러 코드 체계나 컴퓨터 운영체제등에 대해서 동양권이나 다른 문화권의 것도 수용할 수 있도록 하려는 움직임이 있다. 영어 문화권의 경우 보통 7 비트나 8 비트를 가지고 그들의 문자를 표현할 수 있지만 한국, 중국, 일본등을 비롯한 대부분의 동양권 언어는 16 비트 내지는 24 비트를 사용해야만 표현할 수가 있다. 또한 이러한 언어들은 대부분 폰트 화일의 크기가 대단히 크고 문자의 입력 방식도 복잡한 형태를 띄고 있다[12].

현재 한국, 중국, 일본 각국은 제각기 나름대로의 코드 체계를 만들어 사용하고 있는데 이 세 나라의 서로 다른 언어 체계를 포괄할 수 있는 표준화 방안이 마련된다면 이는 컴퓨터에서의 한글 구현과 국제화에 큰 기여가 될 것이다. 따라서 아직 이러한 연구를 구체화하기 위해서는 이미 한글화된 X 위에서 X 툴키트와 그 상위 레벨의 UIL을 포함한 Motif나 OpenLook의 전체에 대한 한글화를 연구하고 중국, 일본등 동양권의 다른 연구와도 상호 협력하여야 한다.

2. 한글화 방안

OSF의 Motif 툴키트를 중심으로 해서 한글화 작업의 방안을 살펴보면 2장에서 기술했던 한글 X의 개발 과정과 같이 크게 3단계의 작업으로 나눌 수 있다. 한글 폰트의 개발 또는 변환, 한글 자동생성기(automata)의 개발, 그리고 각각의 툴키트에 한글 입출력 기능을 추가하는 작업이 필요한데 한글 폰트와 한글 자동생성기는 이미 한글 X 윈도우에서 개발되어 있으므로 세 번째의 작업만을 하면 된다.

2.1 Motif Widget의 분류

widget이란 윈도우나 메뉴등의 리소스와 윈도우의 바탕색이나 커서의 모양과 같은 속성들을 가지고 있는 추상화된 데이타 타입으로 쉽게 말하면 많이 사용할 수 있는 형태의 메뉴나 스크롤바, 윈도우등의 모양이나 속성에 관한 자료 구조와 함수를 미리 만들어 놓은 것이다. 응용 툴을 개발하는 사람들은 이를 이용해서 다양한 형태의 프로그래밍을 할 수가 있다.

Motif에는 기능별로 나누어서 크게 5 종류의 widget이 있다.

○ **Display widget** : 화면에 여러 형태의 입력 영역과 데이타 내용을 출력한다.

○ **Display gadget** : 윈도우를 가지지 않은 Display widget이다.

○ **Container widget** : 응용 툴들에 일반적인

레이아우트의 기능을 제공한다.

○ **Dialog widget** : 사용자와 컴퓨터간의 대화적 접촉을 위한 레이아우트의 기능을 제공한다.

○ **Menu widget** : 여러 형태의 메뉴를 제공한다.

○ **Shell widget** : 가장 높은 레벨의 widget, 윈도우 매니저간의 상호작용을 보호한다.

이러한 widget 각각은 그 하위 레벨에 계층(class)들을 갖고 있으며 각 각의 class는 또 많은 자원을 가지고 있다. class간에는 트리 구조의 상관 관계가 존재하며 하위 레벨의 class에서는 반드시 상위 수준의 class를 포함해야만 표현하고자 하는 대상에 대한 자료 구조를 만들 수 있다.

예컨대 **Dispaly widget**의 하위에는 Core, Primitive, ArrowButton, DrawnButton, Label, List, PushButton, SrollBar, Separator, Text, 그리고 ToggleButton의 11개의 class가 있고, 각 각의 하위에는 여러가지 자원이나 속성들이 있는데 PushButton class를 사용하기 위해서는 반드시 Core, Primitive, Label class를 기반으로 해야만 한다[13].

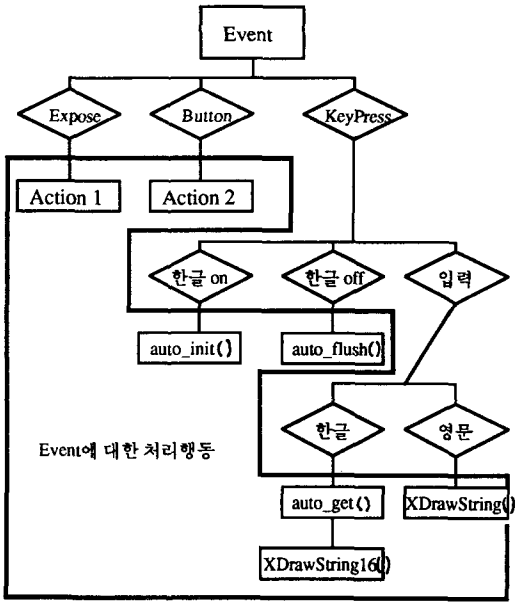
2.2 Widget의 한글화

한글화 작업을 하기 위해서는 모든 widget들에 대한 기능의 분류가 필요하다. 입력 혹은 출력의 기능을 가지고 있는 widget에 한글에 대한 입출력의 기능을 추가하는 것이 주된 작업인데 대부분의 widget은 문자열을 출력하는 기능을 갖고 있다.

한글의 출력 기능을 추가하기 위해서는 X 버전 11 이상에서 제공되는 16 비트 코드 처리 루틴을 사용해서 한글 완성형 2 바이트 코드를 실어나르면 코드 값에 해당하는 한글 폰트를 화면에 출력할 수가 있다. 현재 Motif 툴키트의 출력 부분은 동양권의 16 비트 코드에 대한 처리가 가능하도록 되어 있기 때문에 한글 출력을 처리하는 작업은 비교적 쉽게 할 수 있을 것으로 예상된다.

입력의 기능을 갖고 있는 widget은 **Text widget**, **Dialog widget**, 그리고 **Shell widget** 정도로 정리가 된다. 기존의 widget에 한글 입력을 가능하게 하기 위해서는 먼저 한글 입력 event를 인식할 수 있도록 처리를 해야 한다. event란 마우스의 이동이나 마우스 버튼의 동작 및 키보드 입력등을 말한다.

그림 5-1은 이러한 event를 모아서 처리해주는 **event handler** 루틴의 흐름도이다. 이 흐름도에서 볼 수 있듯이 한글 입력 처리를 가능하게 하기 위해서는 키보드 event에서



< 그림 5-1 > **event handler**의 흐름도

한글 입력의 시작과 끝을 인식할 수 있도록 해야 한다. 한글 입력의 시작을 인식한 순간부터는 한글 자동 *생성기* 루틴을 호출해서 한글 음절의 완성을 결정하고, 화면에는 echo나 local echo를 해준다. 한글 입력의 끝을 인식했을 때는 그때까지 한글 버퍼에 남아있던 자료들에 대한 적절한 처리를 해주고 다시 영문 상태로 돌아가도록 하는 기능을 실현시켜야 한다.

이같은 방법으로 한글 입출력 기능을 구현할 수 있는데 몇 가지 주의해야할 점들이 있다. 대부분 한글 폰트와 영문 폰트의 비율을 2:1의 크기로 하기 때문에 이에 따른 커서의 이동에 대한 적절한 처리가 있어야 하고, 한 음절이 완성되지 않은 상태에서는 커서가 이동을 하지 않도록 한다. 하나의 음절이 행의 끝과 다음 행의 처음에 걸치게 될 때에도 적절한 처리를 해야한다. 이러한 문제는 비단 GUI의 한글화에만 해당하는 것은 아니지만 신경을 써야 할 부분이다.

VI. 맺음말

X 윈도우의 한글화 과정과 GUI 표준화의 추세 및 한글 GUI 개발의 필요성과 방안에 대해서 살펴보았다. 다른 어떤 윈도우 시스템보다도 X가 널리 쓰이고 있고 이를 기반으로 한 GUI들을 표준화하려는 움직임이 거센 만큼, 한글 X 윈도우 시스템의 개발은 X상에서 한글을 사용할 수

있게 되었다는 점과 GUI에 한글 기능을 추가하기 위한 기반을 마련했다는 두 가지 점에서 의미를 갖는다고 할 수 있다.

본 한글 X 윈도우 시스템은 응용 툴들을 개발할 때마다 한글의 입력 생성 과정을 처리해주어야 하는 부담을 안고 있다. 이는 한글의 특수성과 동적인 윈도우의 변화때문에 어느 정도 불가피한 것으로 보이기도 한다. 그러나 X 서버와 같이 시스템 차원에서 한글의 입력을 처리할 수 있다면 응용 프로그래머의 부담이 한결 줄어들 것이다.

이제는 이를 바탕으로 해서 한글 GUI를 개발하는 것이 시급하고 중요한 과제이다. 다행스럽게도 한국, 중국, 일본등 동양 3국의 국력과 기술, 컴퓨터 시장이 무시할 수 없을만큼 커지고 있는 덕분에 한글 기능을 구현할 수 있는 환경이 과거에 비해서 비교적 수월하게 되어가고 있다. 그리고 이 세 나라는 모두 2 바이트의 문자 코드 체계와 한자를 쓰고 있으므로 공통적으로 필요한 부분들에 대한 3국의 협력 작업도 생각해 볼 수가 있다. 이러한 작업은 급속히 발전하고 있는 사용자 접속에 대한 국내적 대처와 국제 표준화를 동시에 해결할 수 있는 길이 될 것이다.

여기에 우리의 손으로 해결해야만 하는 문제점들이 몇 가지 더 있다. 본 한글 X 윈도우 개발의 과정을 통해서 KS-5601 완성형 한글 코드가 규정하고 있는 폰트를 모두 포함한 한글 화일의 크기가 너무 크고 이에 따라 컴퓨터의 응답 속도가 늦어진다는 문제에 집하게 되었다. 성능 개선 작업을 통해서 약간의 조정은 가능하겠지만 이에 대한 근본적인 해결책은 앞으로 풀어야할 문제다. 그밖에도 좀 더 아름답고 깨끗한 한글 폰트를 국내에서 개발하는 문제가 있으며 단순한 한글화의 차원을 넘어선 한국형 운영체제, 한국형 GUI에 대한 연구가 필요하다.

이러한 문제들을 해결하려는 투자와 연구 개발이 있어야 한다는 것을 잊어서는 안 되겠다. 컴퓨터의 목적이 궁극적으로 인간에게 편리함을 주는데에 있다고 한다면, 우리나라에서의 컴퓨터는 우리말을 아는 사람이라면 누구나 쉽게 사용할 수 있는 모습의 것이 되어야 할 것이기 때문이다.

참 고 문 헌

1. A. Nye, "Xlib Programming Manual," Vol. I II, O'Reilly & Assoc., 1988.
2. "OSF Motif Window Manager," OSF, 1989.
3. "OPEN LOOK Graphical User Interface Functional Specification," Sun Microsystems., 1988.
4. R. Scheifler and J. Gettys, "The X Window System," p6, 1986.
5. O. Jones, "Introduction to the X Window System," p2, Prentice Hall, 1989.
6. "X Athena widget," MIT, 1989.
7. "Character Bitmap Distribution Format 2.1," Adobe Sys., 1987.
8. "X Window Manual," MIT, 1989.
9. "한글 한자 정보처리기술 핸드북," 한국표준연구소, 1988.
10. J. Smart, F. Leygues and M. Lichtenhein, "User Interface Technical Assessment Report," p5, 1989.
11. "한글 한자 코드 표준화에 관한 연구," 한국표준연구소, 1987.
12. J. Farrel and D. Lavalee, "User Interfaces in Window Systems," p34, 1989.
13. "OSF Motif Toolkit," OSF, p3-2, 1989.
14. "UNIX System V Rel. 4.0 Software Developer Conference '88," AT&T Unix Pacific, p747, 1989.
15. "표준화를 향한 유저 인터페이스," 경영과 컴퓨터 7월호, 1989.
16. 조명진, 하종성, "한글 X 윈도우 시스템 Manual," Integration Project Technical Report, 1989.