특 집

임베디드 그래픽 윈도우 시스템 기술

김 성 우*, 이 경 우**

● 목 차 ●

1. 서 론

임베디드 시스템은 수십 년 전부터 서버, 공정 제어 시스템 등의 다양한 용도로 개발되어 왔다. 컴퓨터 그래픽 장치들이 미비했던 초기의 임베디드 시스템에서는 그래픽 사용자 인터페이스가 아예 없거나 아주 단순하게 구현되었었다. 컴퓨터 하드웨어 기술의 발달과 응용 프로그램들이 복잡해짐에 따라, 사용자가 더욱 쉽게 임베디드 시스템과 접속할 수 있기를 바라는 요구가 점차 증대되었다. 하지만, 80년대 말까지는 그래픽 하드웨어와 입출력 장치들의 성능 제한으로 그래픽 사용자 인터페이스는 여전히 텍스트 기반이나 흑백 화면을 이용한 것들이 대부분이었다.

90년대 들어 그래픽 하드웨어 및 입출력 장치들의 성능이 더욱 우수해지고 인터넷 기술이 보편화됨에 따라, 홈서버, PDA, 웹패드 등의 다양한 정보 단말 기기들이 등장하였다. 이러한 임베디드 장치들은 범용 컴퓨터 사양에 못지 않은 화려한 컬러 화면 장치들과 터치스크린, 마우스, 무선 키보드 등의 다양한 입력 장치들을 지원할 수 있게 되었다. 한편, 기존의 VxWorks, QNX, Nucleus 등의 기존의 실시간 운영체제 외에도 윈도우즈 CE, 임베디드 리눅스 등의 운영체제들이 점차 임베디드 시스템에 적용되고 있다[1]. 이처럼 다양한 임베디드 장치들과 이를 구동시키는 운영체제 환경의 등장은 종래의 범용 그래픽 윈도우 시스템의 임베디드화는 물론, 소형 임베디드 그래픽 윈도우 시스템의 기술 개발을 가속시키고 있다.

본 고에서는 임베디드 그래픽 시스템에서의 요구사항, 시스템의 전체적인 구조 및 새로운 그래픽 처리 기법들, 그리고 현재 실제로 구현된 임베디드 그래픽 윈도우 시스템 사례에 대해 살펴본다.

2. 시스템 요구사항

한정된 용도로 쓰이는 임베디드 시스템의 특성에 따라 일반적으로 범용 그래픽 윈도우 시스템을 임베디드 용으로 사용하기에는 많은 무리가 따른다. 그러므로, 임베디드 그래픽 윈도우 시스템은 개발시 범용 그래픽 윈도우 시스템과는 다른 요구조건들을 우선적으로 고려하여야 한다.

먼저, 임베디드 시스템은 일반적으로 정해진 용

* ETRI 컴퓨터소프트웨어기술연구소 선임연구원
** ETRI 컴퓨터소프트웨어기술연구소 연구원

도에 맞게 최적화되어 설계되므로, 메모리 등의 시스템 자원이 제한적이다. 이런 제한된 조건, 특히 최소량의 메모리를 이용하여 경량의 그래픽 윈도우 시스템을 구축하여야 한다. 통상, 임베디드 그래픽 윈도우 시스템은 라이브러리를 포함하여 수 메가 바이트 이하의 footprint를 가지게 된다. 또한, 임베디드 시스템은 범용 시스템에 비해 더욱 고장에 치명적이다. 특히, 대규모 제어 시스템, 정보가전 등의 임베디드 시스템에서의 안정성은 아무리 강조해도 지나치지 않다. 그러므로, 임베디드 그래픽 윈도우 시스템도 내고장성 및 안정성을 높이도록 충분한 검증을 거쳐야 한다.

현재 임베디드 시스템 용으로 사용되고 있는 각종 CPU 및 고성능 그래픽 카드, 터치스크린, 마우스, 리모콘 등과 같은 입출력 장치들은 다양한 쓰임새와 하드웨어 방식으로 구현되어 있다. 그러므로, 임베디드 그래픽 윈도우 시스템은 이런 다양한 하드웨어 장치들을 지원하기 위해서는 한정된 시스템 자원 하에서 자유롭게 이식하거나 제거할 수 있는 이식성 높은 구조를 가져야만 한다.

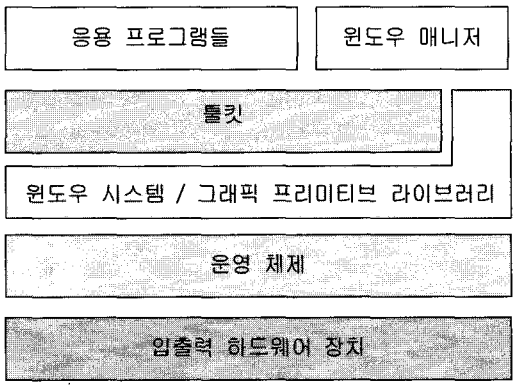
임베디드 그래픽 윈도우 시스템은 소프트웨어적인 관점에서 특정한 응용 프로그램에서 요구하는 모듈들을 선택하여 그래픽 사용자 인터페이스를 재구성할 수 있어야 한다. 특히, 폰트 및 그래픽 파일 처리 모듈은 사용자가 원할 때 추가 및 삭제가 가능하도록 하는 것이 필요하다.

그 밖에, 응용 프로그램 개발자들이 개발 기간을 단축할 수 있도록 호스트 개발 환경에서 쉽게 그래픽 사용자 인터페이스를 구성하거나 시뮬레이션할 수 있는 개발도구를 제공하는 부가적인 요구사항을 가질 수 있다.

3. 시스템 구조

응용 프로그램의 그래픽 사용자 인터페이스는 소프트웨어 사용자가 보고 접속하는 부분이다. 이러한 인터페이스의 "look and feel"을 구현하는 대표적인 방법은 그래픽 화면을 여러 영역의 "윈도우"로 나누는 윈도우 시스템이다[2,3]. 윈도우 시스템은 일반적으로 다음과 같은 구조를 가진다.

그래픽 윈도우 시스템은 (그림 1)과 같이 크게 몇 가지 부분으로 나뉜다. 먼저 윈도우 매니저는 사용자가 윈도우 생성, 크기 재설정, 이동 등과 같은 요청으로 그래픽 윈도우 시스템과 접속하는 사용자 인터페이스 부분을 담당한다. 윈도우 시스템은 그래픽 프리미티브 라이브러리를 통해 글자, 점, 선, 사각형 등의 기본적인 2D 프리미티브들을 그래픽 화면에 그리거나 특정 영역을 화면의 다른 영역으로 복사하며, 입력 하드웨어 장치들을 통해 사용자로부터의 입력을 얻어서 윈도우 생성, 크기 재설정, 이동시키는 실제적인 그래픽 처리를 담당한다. 툴킷은 록 메뉴, 버튼, 스크롤 바 등의 많은 유용한 위젯들을 가진 고수준의 라이브러리로서 다양한 응용 프로그램들에게 최적의 프로그래밍 인터페이스를 제공한다. 각각의 구성 요소들은 그래픽 응용 프로그램 개발 시 사용할 수 있는 최적의 프로그래밍 인터페이스를 제공하여야 한다.



(그림 1) 그래픽 윈도우 시스템 구조도

보통 윈도우 매니저는 (그림 1)에서처럼 윈도우 시스템 상위에 두는 하나의 클라이언트 프로그램이다. 하지만, 서로 다른 look and feel을 가지는 다

중 윈도우 매니저들을 필요로 하지 않는 소형의 임베디드 그래픽 윈도우 시스템에서는 윈도우 매니저와 윈도우 시스템의 구별을 따로 하지 않는 구조를 가진다.

4. 새로운 그래픽 처리 기법

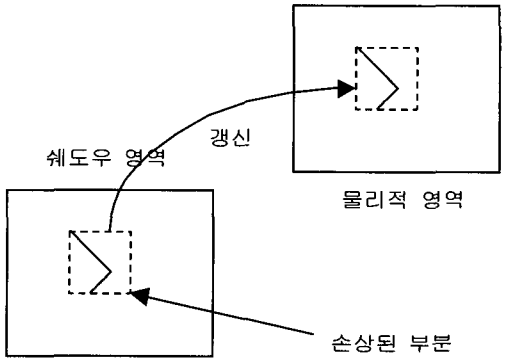
본 장에서는 점차 증가하는 이동성 정보단말 기기 등의 임베디드 시스템 환경에서 유용한 새로운 그래픽 윈도우 처리 기법들에 대해 소개한다.

4.1 동적 화면 크기 재설정(Resize) 및 회전(Rotation) 기법

핸드헬드(handheld) 컴퓨터나 PDA 과 같은 포터블 컴퓨터에서 전자북과 같은 응용 프로그램에 따라 화면 크기를 바꾸거나 회전시키는 기능이 요구되었는데, 이를 구현한 것이 동적 화면 크기 재설정 및 회전 기법이다[4]. 다만 응용 프로그램이나 툴킷 수준이 아닌, 서버 수준에서 전체 화면의 크기 변화 또는 회전을 수행하여야 한다. 또한 화면 크기가 재설정되거나 회전되었을 때 클라이언트 프로그램들에 알리는 기능도 가질 수 있다.

동적 화면 크기 재설정 기법은 이전 화면에 대한 렌더링을 중지하고 새로운 화면 크기로 그래픽 장치를 재설정하고 렌더링을 다시 동작시킴으로써 구현할 수 있다. 화면 회전 기법은 두 가지 방법으로 구현할 수 있는데, 첫째는 모든 렌더링 함수들에 회전 기능을 추가하는 방법이고, 또 다른 방법은 쉐도우 프레임버퍼(shadow framebuffer) 를 이용하는 것이다. 쉐도우 프레임버퍼란 주 기억 장치에 할당된 일종의 가상 프레임버퍼를 말한다. (그림 2)는 쉐도우 프레임버퍼의 동작을 나타낸다.

예컨대 X 윈도우에서는 Xrandr 확장 라이브러리를 통해 가상 프레임버퍼를 이용하여 시스템을 재부팅하지 않더라도 동적으로 클라이언트에서 X 윈도우 화면의 크기를 바꾸거나 회전시킬 수 있도록 구현하였다.



(그림 2) 쉐도우 프레임버퍼의 동작

4.2 이동(Migration)과 복제(Replication) 기법

임베디드 그래픽 윈도우 시스템에서 이동이란 기존의 실행되는 응용프로그램이 다른 장치로 디스플레이를 전환하는 것을 말한다[4]. 복제는 같은 응용 프로그램이 다중 장치들 위에 그 출력을 중복시키는 것을 말한다[4]. 사용자들이 핸드헬드 컴퓨터, 데스크탑, 정보가전 등과 함께 이동하고 연결할 때 그래픽 응용 프로그램들은 각각의 디스플레이들 사이에서 이동할 수 있는 기능이 필요하다. 더구나 이 응용 프로그램들은 서버와의 연결이 끊어졌을 때도 살아남을 수 있어야만 한다. 그것들은 나중에 다시 그런 장치들과 재연결할 수 있어야 한다.

이동과 복제는 다양한 픽스맵(pixmap) 및 프레임버퍼 depth, pseudocolor 화면에 대한 지원을 보장하여야 하므로 실제로 구현하기 어렵다. 가장 좋은 해결책은 응용 프로그램들과 툴킷들이 이동과 복제 기능을 지원하는 것이다. 다행히 pseudocolor 디스플레이들은 지난 10년간 점차 사용되지 않고 대신 truecolor 디스플레이를 대부분의 하드웨어가 지원하게 되었으며, 좋은 그래픽 윈도우 시스템은 다양한 색 데이터들 사이를 자동적으로 변환해 주게

되었다. 또한 툴킷들이 윈도우 시스템으로부터 응용프로그램들을 완전히 독립시키고, 서버 이동과 복제를 가능하도록 적용하기 쉽게 되어야 한다.

4.3 영상 혼합(image composition) 기법

영상 혼합이란 영상 데이터를 색 비트들로 표현함에 따라, 렌더링의 핵심 라스터 처리(raster-op)를 영상 혼합 처리기들로 대체한 것이다[5]. 이 처리기들은 투명성(transparency)을 보장하고, 영상들이 서로의 위에 그려진 것처럼 혼합함으로써 자연스러운 방법으로 색 데이터를 처리한다. 이 기법은 또한 anti-aliasing을 근사화하기 위해서도 이용할 수 있다.

영상 혼합 처리의 한 방법은 한 영상을 다른 영상 위에 두어 위에 놓은 영상의 투명한 영역을 통해 아래에 놓인 영상이 보이도록 하는 것이다. 픽셀 값의 흐릿한 정도를 나타내는 "alpha"를 0과 1 사이의 도입함으로써 간단한 방정식으로 두 픽셀 색을 함께 표현할 수 있다. 이와 같은 알파 혼합 방식은 다중 영상을 혼합하기 위하여 그래픽 처리 하드웨어에서 많이 쓰이는 방식이기도 하다. 다음 식을 "over" 처리기라고 한다.

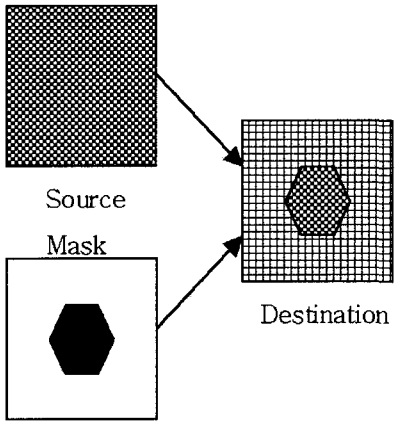
$$C_{\text{result}} = C_{\text{under}}(1 - \alpha_{\text{over}}) + C_{\text{over}}\alpha_{\text{over}}$$

또다른 처리 방법은 한 이미지를 다른 이미지로 마스크하는 것이다. 마스크의 투명한 영역은 그 이미지로부터 없어지고 마스크의 흐릿한 영역들은 이미지를 보이게 한다. 이것을 "in" 처리기라고 한다.

$$C_{\text{result}} = \alpha_{\text{mask}} C_{\text{image}}$$

최종적인 이미지 혼합 기법들을 기반으로 한 통합된 렌더링 처리는 다음 식으로 표현되고, (그림 3)과 같이 동작된다.

$$C_{\text{result}} = (C_{\text{image}} \text{ IN } C_{\text{mask}}) \text{ OVER } C_{\text{result}}$$



(그림 3) 통합 영상 혼합의 동작

영상 픽셀 렌더링에서의 모든 처리는 기본적인 혼합 처리기들에 대해 정해지므로 일관된 모델을 제공하고 최소한으로 구현되어야 하며, 가능한 한 쉬운 방향으로 윈도우 시스템에 내재된 처리 방법들을 제공함으로써 툴킷 및 응용 프로그램들과 잘 동작하도록 설계되어야 한다.

5. 시스템 사례

본 장에서는 임베디드 리눅스 환경에서 주로 사용되는 임베디드 그래픽 윈도우 시스템의 몇 가지 사례를 들어 설명한다. 이 외에도 상용으로 사용되는 그래픽 윈도우 시스템으로는 VxWorks용 UGL/Zinc 그래픽 윈도우 라이브러리, 윈도우즈 CE 및 윈도우즈 XP 임베디드용 그래픽 윈도우 시스템 등이 있다.

일반적인 데스크탑 리눅스에서 많이 사용하는 GNOME, KDE 등의 그래픽 윈도우 시스템은 최소한 10 메가 바이트 이상의 footprint를 가지므로, 그대로 임베디드 환경으로 사용하기 힘들다. 그 대안으로서 다양한 오픈 소스 프로젝트들을 통하여 많은 연구가 이루어지고 있는데, GtkFB, Qt/Embedded, Microwindows, ViewML 등을 들 수 있다. 이들 임베디드 그래픽 윈도우 시스템은 여타

상용 임베디드 그래픽 윈도우 시스템에 비해 이식 환경 및 개발 도구들이 미비한 단점이 있지만, 기존의 데스크탑에서 동작하는 수많은 공개 소스의 응용 프로그램들을 거의 그대로 이용하거나 로열티가 없는 등의 많은 잇점을 가진다.

5.1 GtkFB

GTK+ 는 본래 GIMP라 불리는 영상 처리 프로그램의 그래픽 사용자 인터페이스 환경을 위한 툴킷으로 제작된 GNU 프로젝트였다[6]. 하지만, 이렇게 만들어진 GTK+ 그래픽 툴킷은 확장성, 이식성, 안정성이 뛰어나서 기존의 상용 X 윈도우 툴킷을 대체하여 점차 인기를 얻게 되었다. 하지만, 다양한 언어와 위젯들을 지원하는 반면 thread 기반의 소형 임베디드 용으로 쓰기에는 크기가 여전히 큰 편이었으므로, 하부의 X 윈도우 대신 직접 프레임버퍼에 그래픽 처리를 하도록 구현한 것이 GtkFB 이다. GtkFB는 상대적으로 크기가 작은 대신, 다양한 그래픽 장치 가속기들을 지원하지 않으며 단일 프로세스 모델로 구현된 단점을 가진다[7].

5.2 Qt/Embedded

Qt는 trolltech 사에서 개발한 C++ 기반 상용 GUI 툴킷이다. 이것은 사용자에게 고급 위젯들을 많이 제공하고, 국제화 및 지역화가 가능하도록 다중 언어를 지원하며, 리눅스는 물론 MS 윈도우즈 등의 다양한 플랫폼에 이식 가능한 특징을 갖는다. 또한, QPE(Qt Palmtop Environment)라 불리는 개인 정보 관리 패키지는 물론, 우수한 개발 도구들을 제공한다. Qt/Embedded는 GtkFB처럼 소형 임베디드 용으로 사용하기 위하여 하부의 X 윈도우를 쓰지 않고 프레임버퍼에 직접 접근 가능하도록 수정한 것이다[8]. Qt는 라이브러리 소스는 공개하고 있으나, Qt를 이용한 개발 결과를 상용화할 시 자사에서 정한 라이센스 정책을 따라야만 한다.

5.3 Microwindows

Microwindows는 Century Software 사가 임베디드 시스템용으로 개발한 오픈 소스 그래픽 윈도우 시스템이다. 앞서 예를 든 GtkFB나 Qt/Embedded처럼 하부의 X 윈도우를 들어내고 Nano-X 서버 및 프레임버퍼용 그래픽 프리미티브 라이브러리를 구현하였다. 또한, Microwindows 는 QPE처럼 PDA 환경도 제공하고 있다[9].

6. 결 론

본 고에서는 다양한 임베디드 시스템에 탑재되어 사용자에게 그래픽 사용자 인터페이스를 제공하는 임베디드 그래픽 윈도우 시스템에서의 요구사항, 전체적인 구조와 기능, 현안 그래픽 처리 기술들과 실제 사례들에 대해 살펴보았다. 특히 기존의 정형화된 데스크탑 방식과는 다른 새로운 기법들이 임베디드 시스템에서 많이 요구된다는 점에서 그 중요성이 부각되고 있다.

인터넷과 핸드폰을 지나 포스트 PC 시대에 접어들면서 고급의 데스크탑 그래픽 윈도우 시스템 기술들이 임베디드 환경에 점차 적용되고 있고 다양한 정보단말을 포함한 임베디드 시스템들이 개발되고 있는 실정이다. 빠르게 변화하는 임베디드 시장에서 효과적으로 대응하기 위하여 국내에서도 임베디드 그래픽 윈도우 시스템에 관한 표준화 및 지속적인 기술 개발에 많은 노력을 기울여야 한다.

참고문헌

[1] Wayne Wolf, "What Is Embedded Computing?", IEEE Computer, pp 136-137, January, 2002.

[2] Foley, et al., Computer Graphics: Principles and Practice, 2nd edition, Addison-Wesley, 1996.

[3] Brad A. Myers, "User Interface Software Technology", ACM Computing Surveys, Vol. 28,

No. 1, March 1996.

[4] Jim Gettys, Keith Packard, "The X Resize and Rotate Extension - RandR", 2001 Usenix Annual Technical Conference, Boston, MA, June 2001.

[5] Keith Packard, "Design and Implementation of the X Rendering Extension", 2001 Usenix Annual Technical Conference, Boston, MA, June 2001.

[6] Havoc Pennington, GTK+/GNOME Application Development, New Riders Publishing, 1999.

[7] Alexander Larsson, "GTK+ for the Linux Framebuffer", RedHat white paper, http://www.redhat.com/devnet/articles/gtkfb-whitepaper/, 2001.

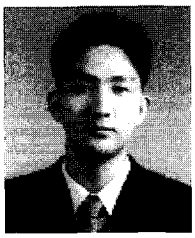
[8] "Qt/Embedded: The C++ embedded GUI application developer's Toolkit", Trolltech white paper, ftp://ftp.trolltech.com/qt/pdf/QtEWhitepaper.pdf, 2001.

[9] The Microwindows Project, http://microwindows.org

저자약력

김 성 우

1991년 한국과학기술원 전기 및 전자공학과(공학사)
1993년 한국과학기술원 전기 및 전자공학과(공학석사)
1999년 한국과학기술원 전기 및 전자공학과(공학박사)
1999년-현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 선임연구원
관심분야 : 실시간 OS, 그래픽 윈도우 시스템, 내고장성 시스템
e-mail : kimsw@etri.re.kr



이 경 우

1999년 고려대학교 전기공학과(공학사)
2001년 한국과학기술원 전기 및 전자공학과(공학석사)
2001년-현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 연구원
관심분야 : 실시간 OS, 실시간 시스템, 통신프로토콜, 임베디드시스템
e-mail : kwlee892@etri.re.kr