임베디드 시스템을 위한 그래픽 가속 모듈에 관한 연구

김성우*, 설동명**
동의대학교 컴퓨터소프트웨어공학부*, 한국전자통신연구원**
e-mail : libero@dongeui.ac.kr*, dmseol@etri.re.kr**

A Study on a Graphic Acceleration Module for Embedded Systems

Seong-Woo Kim*, Dong Myung Seol**
Division of Computer and Software Engineering, Dong-Eui University*
Embedded GUI Research Team, Embedded Software Center, ETRI**

요 약

임베디드 시스템을 위한 그래픽 윈도우 시스템은 특정 기능을 가진 시스템의 사용자에게 편리한 그래픽 사용자 인터페이스 환경을 제공하여 준다. 이러한 시스템에서 그래픽 처리 성능을 본질적으로 향상시키기 위해서는 그래픽 하드웨어가 제공하는 선, Bitblt, 패턴 채우기 등의 2D 그래픽 가속 기능을 이용한 그래픽 가속 모듈의 구현이 필수적이다. 본 논문에서는 임베디드 시스템을 위한 오픈 윈도우 환경인 Tiny X 에서 그래픽 가속 모듈을 구현하는 방법을 상세히 제시한다. 뿐만 아니라, 제시한 방법대로 구현된 그래픽 가속 모듈의 성능 평가를 통하여 그 효율성을 검증하였다.

1. 서론

Xerox 가 획기적으로 고안해낸 이후로 그래픽 사용자 인터페이스 (GUI) 는 컴퓨터의 핵심적인 부분으로 꾸준히 사용되어왔다. 근래에 와서, 다양한 입출력 장치와 고성능 프로세서 등의 컴퓨터 하드웨어 및 네트웍 기술의 발전으로 범용의 개인용 컴퓨터 중심으로 사용되던 화려한 그래픽 처리가 임베디드 시스템 용으로 서서히 확산되고 있는 추세이다.

임베디드 시스템 용 그래픽 윈도우 시스템은 제한된 메모리 자원 등을 고려하여 수 메가 바이트 이하의 작은 footprint 를 가지며, 경량이어야 하고, 다양한 입출력 장치들이나 특정 모듈을 쉽게 추가하거나 제거할 수 있는 재구성 능력을 가져야 한다. 그러므로, 이처럼 제한된 환경에서도 원하는 그래픽 처리 성능을 가지기 위해서는 그래픽 하드웨어가 제공하는 그래픽 가속 기능 등을 충분히 활용하여야 한다[1].

임베디드 시스템 용 그래픽 윈도우 시스템으로는 Windows CE 와 상용 RTOS 에 주로 쓰이는 상업적 용도의 GUI 들과 임베디드 리눅스 등에 주로 무료로 쓰이는 것 등 여러 종류가 있는데, 임베디드 리눅스 용으로는 Qt Embedded, Microwindows, Tiny X 등이 있다. 본 논문에서는 무료로 사용할 수 있고, 기존의 X 윈도우 시스템의 기능을 거의 그대로 사용할 수 있는 Tiny X 에 대하여 상세하게 설명하고, 하부의 그래픽 가속 모듈을 구성하는 방법에 대해 살펴본다. 제시된 그래픽 가속 모듈을 사용하는 방법과 그렇지 않은 방법에 대한 성능 비교를 통하여 그 효율성을 검증할 수 있다.

2. 임베디드 그래픽 윈도우 시스템

일반적으로 그래픽 윈도우 시스템은 내부적으로 다음 그림과 같은 계층 구조를 가진다[2]. 윈도우 매니저는 사용자가 윈도우 생성, 크기 재설정, 이동 등과 같은 요청으로 그래픽 윈도우 시스템과 접속하는 사용자 인터페이스 부분을 담당한다. 그래픽 윈도우 라이브러리는 그래픽 장치 구동기를 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 메뉴, 버튼, 스크롤 바 등의 많은 유용한 위젯들을 가진 고수준의 라이브러리로서 다양한 응용 프로그램들에게 최적의 프로그래밍 인터페이스를 제공한다.

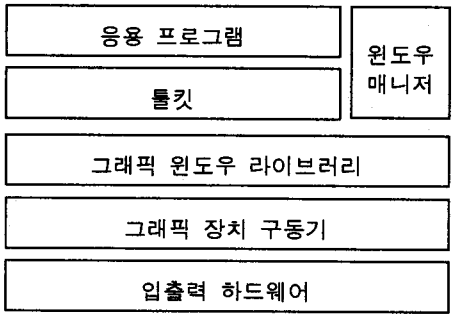

그림 1. 그래픽 윈도우 시스템 구조도

특히, 윈도우 매니저는 그래픽 윈도우 라이브러리 상위에 두는 하나의 클라이언트 프로그램이다. 네트웍을 통한 서버-클라이언트 구조로 구현되어 다중 프로그램 디스플레이가 가능하며 다른 컴퓨터로의 출력도 가능하다. 하지만, 네트웍 오버헤드나 서로 다른 look and feel 을 가지는 다중 윈도우 매니저들을 필요로 하지 않는 소형 임베디드 그래픽 윈도우 시스템에서는 윈도우 매니저와 그래픽 윈도우 라이브러리의 구별을 따로 하지 않는 구조를 가지기도 한다.

2.1 그래픽 하드웨어의 2D 가속 기능

선이나 사각형 등의 2D 프리미티브, 또는 특정 그래픽 영역을 다른 영역으로 복사할 때 보다 빨리 처리하는 그래픽 가속 기능은 그래픽 하드웨어 자체 구조에 의해 보통 구현된다. 다음은 일반적으로 그래픽 하드웨어에서 구현할 수 있는 2D 그래픽 가속 기능들을 나열한 것이다[3].

BitBlt	특정 크기의 소스 윈도우를 목표 윈도우로 복사하는 기능
SolidFill	특정 크기 윈도우를 색으로 채우는 기능
SolidLine	특정한 색/길이의 얇은 선을 그리는 기능
Color Expand Fills	모노 비트맵으로부터 색 확장 사각형 (color expansion rectangle) 을 채우는 기능. 1. opaque 모드 (비트맵에서 0 은 배경색) 2. 투명 모드 (비트맵에서 0 은 그대로)
8x8 Pattern Fills	8x8 패턴은 오프스크린 메모리에 연속으로 저장된 8x8=64 개 픽셀을 가진 컬러/모노 패턴. 한 번 오프스크린 메모리에 저장된 컬러/모노 패턴은 픽스맵 캐쉬로 사용될 수 있어, 다음 패턴 채움 시에 시간 절약 1. 컬러 패턴을 채우는 tiling 기능 2. opaque stipple 채우는 기능 3. 투명 stipple 채우는 기능

3. Tiny X 그래픽 장치 구동기

Tiny X 는 Keith Packard 에 의해 개발된 임베디드 시스템 용 X 서버이다[1,4]. 그러므로, 작은 메모리 footprint 에서도 잘 동작하도록 기존의 표준 Xfree86 서버의 하부를 대폭 수정하고 줄여 작은 footprint 를 갖도록 구현하였다. 특히, 표준 Xfree86 서버가 지원하는 모듈 기능을 없앴으며, 지원하는 X 확장 기능도 한정되어, 일반적인 tiny X 서버의 크기는 x86 CPU 에서 수행할 때 1MB 정도의 크기를 가진다.

현재 Tiny X 는 1,4,8,16,24,32 bpp(bit per pixel) 흑백/칼라 모드를 실행하는 기존의 cfb/mfb 보다 더 작은 새로운 프레임버퍼 코드를 사용한다. 기본적인 tiny X 서버는 프레임버퍼 또는 VESA 표준 그래픽 입출력에 기반한 Xfbdev 서버와 Xvesa 서버 2 가지가 있으며, 기타 igs, mach64, iPAQ, trident, pcmcia 등의 그래픽 보드를 위한 서버를 지원한다.

3.1 데이터 구조

Tiny X 그래픽 장치 구동기는 간단한 데이터 구조를 제공하는데, 카드 정보 구조체와 스크린 정보 구조체를 설계하는 것이 필요하다. 카드 정보 구조체는 그래픽 카드가 디폴트로 제공하는 드라이버(vesa, fb 등), 그래픽 레지스터들 (vga 등의 일반 그래픽 레지스터들 및 기타 전용 그래픽 레지스터들), 그래픽 카드의 IO 및 프레임버퍼 어드레스, 프레임버퍼 크기 등의 정보를 가진다. 스크린 정보 구조체는 하드웨어 커서를 사용할 경우 관련된 정보들과 그래픽 가속을 위한 오프스크린(offscreen) 패턴 및 타일 정보 등을 포함할 수 있다.

3.2 기본 기능

TinyX 서버가 실행되면 호출되는 디바이스 수준의 stub 루틴은 크게 세 가지인데, 그래픽 장치 구동기의 일반 루틴들을 등록하는 InitCard() 함수, 출력 장치 구동기를 초기화하는 InitOutput() 루틴, 입력 장치 구동기를 초기화하는 InitInput() 루틴이다.

그래픽 장치 구동기의 주요 모듈은 KdCardFuncs 이라는 구조체의 등록 함수들인데, 다음과 같다.

그래픽 카드 설정 함수
Bool (*cardinit) (KdCardInfo *); /* 장치검색과 메모리매핑 */ Bool (*scrinit) (KdScreenInfo *); /* ScreenInfo 구조체초기화 */ Bool (*initScreen) (ScreenPtr); /* ScreenRec 초기화 */ void (*preserve) (KdCardInfo *); /* 그래픽카드 상태 저장 */ Bool (*enable) (ScreenPtr); /* 그래픽 렌더링모드 설정 */ Bool (*dpms) (ScreenPtr, int); /* DPMS 전원관리모듈설정 */ void (*disable) (ScreenPtr); /* 렌더링 모드를 끔 */ void (*restore) (KdCardInfo *); /* 그래픽카드 상태 복구 */ void (*scrfini) (KdScreenInfo *); /* 스크린을 닫음 */ void (*cardfini) (KdCardInfo *); /* 그래픽 카드 해제 */
하드웨어 커서 함수들

Bool (*initCursor) (ScreenPtr); /* 하드웨어커서 검색초기화*/ void (*enableCursor) (ScreenPtr); /* 하드웨어 커서 enable */ void (*disableCursor) (ScreenPtr);/* 하드웨어 커서 disable */ void (*finiCursor) (ScreenPtr); /* 하드웨어 커서를 끔 */ void (*recolorCursor) (ScreenPtr, int, xColorItem *); /* 하드웨어 커서 recolor */
그래픽 가속 함수들
Bool (*initAccel) (ScreenPtr); /* 그래픽가속 기능 초기화 */ void (*enableAccel) (ScreenPtr); /* 그래픽가속 기능 enable */ void (*syncAccel) (ScreenPtr); /* 그래픽가속 sync 루틴 */ void (*disableAccel) (ScreenPtr); /* 그래픽가속기능 disable */ void (*finiAccel) (ScreenPtr); /* 그래픽가속 기능 끔 */
pseudo color 색 설정 함수들
void (*getColors) (ScreenPtr, int, int, xColorItem *); /* pseudo color 얻음 */ void (*putColors) (ScreenPtr, int, int, xColorItem *); /* pseudo color 설정 */
스크린 재설정 함수들
Bool (*finishInitScreen) (ScreenPtr); /* ScreenRec 를 끝냄 */

3.3 그래픽 가속 기능

기본 Xfbdev 와 Xvesa Tiny X 서버들은 그래픽 하드웨어 자체 내의 가속 기능을 이용하지 않고 소프트웨어적으로 2D 그래픽 처리를 수행하므로 고속 그래픽 처리를 요구하는 응용을 감당하기 어렵다. 다행히, 표준 Xfree86 서버 소스에 포함된 많은 그래픽 하드웨어 장치 구동기들의 소스코드를 이용하여 Tiny X 그래픽 가속 모듈을 구현하는 것이 가능하다.

Tiny X 그래픽 장치 구동기의 그래픽 가속 기능은 기존의 표준 Xfree86 서버의 그래픽 가속 구조 XAA (X Accerelation Architecture) 의 구조 및 기능을 대폭 간소화된 KAA (Kdrive Acceleration Architecture) 를 이용하여 구현한다[3]. 가속 기능은 주로 스크린 상의 윈도우와 같은 drawable 이나 그래픽 컨텍스트(GC) 를 통하여 구현할 수 있다.

1. ScreenRec 에 등록되는 그래픽 가속 함수들

ScreenRec 에 등록되는 함수들 중 CreateGC 를 제외하고는 스크린 윈도우 단위로 그래픽 가속 기능을 수행한다[5]. 대표적으로, CopyWindow() 함수는 윈도우를 이동하였을 때 실행되는 함수이고, PaintWindowBackground() 와 PaintWindowBorder() 함수는 윈도우의 배경색 또는 배경 그림, 테두리 등을 그리는 기능이다[5,6].

CopyWindow	윈도우를 이동하였을 때 실행되는 함수
PaintWindowBackground	윈도우의 배경색 또는 배경그림 등을 그리는 함수. None, ParentRelative, BackgroundPixmap, BackgroundPixel 의 네 모드로 나뉘어져 실행
PaintWindowBorder	윈도우의 테두리를 그리는 함수.

2. 그래픽 컨텍스트를 통한 그래픽 가속 함수들

그래픽 컨텍스트를 통해 구현되는 그래픽 가속에 관련된 함수들은 CreateGC() 함수에 의해 생성된 GC 와 GCFuncs 구조체에 의해 GCOps 구조체로 등록되어 그래픽 컨텍스트(GC) 내에서 구현된다[5,6]. 자세한 함수 리스트는 다음과 같다.

CopyArea	한 프레임버퍼 영역을 다른 위치 영역으로 복사하는 기능.
CopyPlane	소스 drawable 로부터 목표 drawable 로 사각영역의 한 plane 을 복사
PolyPoint	여러 점들을 그리는 함수
PolyLines	여러 연결된 선들을 그리는 함수
PolySegment	연결되지 않은 선들을 그리는 함수
PolyRectangle	여러 사각형을 그리는 함수
PolyArc	여러 원호를 그리는 함수
FillPolygon	다각형을 채우는 함수
PolyFillRect	여러 사각형을 채우는 함수
PolyFillArc	여러 원호를 채우는 함수
PutImage	픽스맵 이미지를 한 drawable 로 복사하는 함수
PutText	문자들을 그리는 함수. 글자를 채우는 모양과 8/16bit 글자 코드에 따라 ImageText8, ImageText16, PolyText8, PolyText16 로 나뉘어짐.
FillSpans	적절한 패턴, stipple 로 픽셀의 수평 열들을 채우는 함수.
GlyphBlt	문자들을 그리는 요청에 대한 개별적인 글자 glyph 들을 그리는 함수
PushPixels	색, 타일, stipple 등을 나타내는 비트맵으로 정의된 어떤 형태로 drawable 의 전경, 배경 등을 채우는 함수

4. 성능 평가

본 논문의 성능 평가를 위해 VIA 사의 Mini-ITX M10000 보드를 사용하였다. 이 보드에는 64 비트 VIA C3 프로세서 기반의 데스크탑 PC 를 위한 고성능 저가격 에너지 효율적인 VIA CLE266 그래픽 칩셋을 내장하고 있다. CLE266 에 포함된 VT8623 노스브릿지는 128 비트 그래픽 가속기를 포함하여 2D, 3D, 그리고 DVD Video 가속 기능을 구현하고 있다. 본 연구에서는 표준 X 서버용 VIA 장치 구동기를 참조하여 VIA CLE266 그래픽 칩셋을 위한 Tiny X 서버용 2D 가속 모듈을 구현하였다.

2D 그래픽 가속 모듈 성능을 평가하기 위해 X 윈도우 유틸리티 프로그램인 x11perf 를 사용하였다. x11perf 는 X 윈도우 서버가 다양한 크기 (1x1, 10x10, 100x100, 500x500 등) 의 사각형, 선, 원, 텍스트 그리기, 스크롤, 윈도우 복사 등의 기능을 수행하여 초당 반복수행한 평균 결과를 출력한다[7]. 다음 그림은 x11perf 를 사용하여 그래픽 처리 기능을 수행하는 모습을 캡처한 것이다.

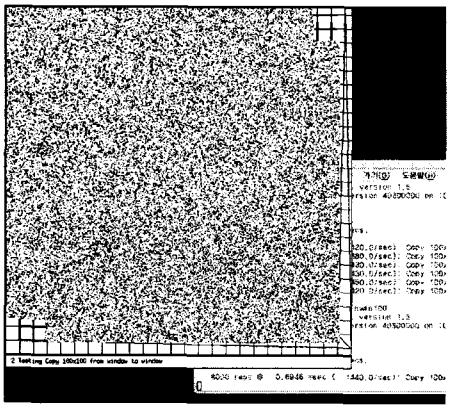

그림 2. x11perf 의 실행 화면

다음 그림들은 2D 그래픽 가속 기능을 사용한 TinyX 서버 (Xvia) 와 사용하지 않는 vesa 모드 TinyX 서버 (Xvesa) 를 실행시킨 상태에서, x11perf 를 각각 수행한 결과를 나타낸다. 각각의 그래픽 처리 기능에 대한 y 축의 단위는 초당 반복수행횟수 를 나타낸다.

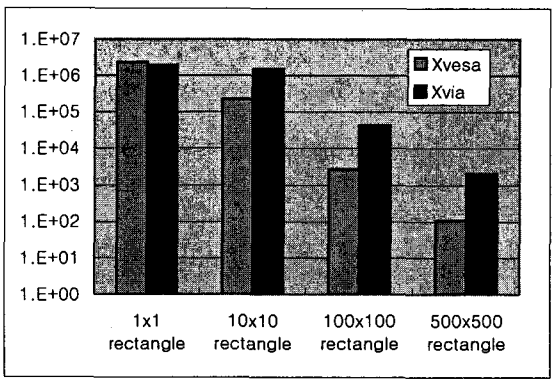

그림 3. 사각형 그리는 함수에 대한 성능

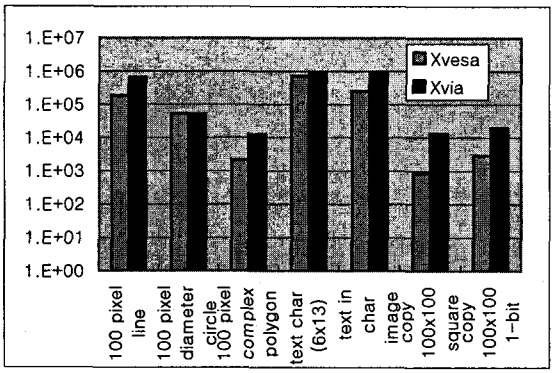

그림 4. 여러 그리기 함수들에 대한 성능

사각형을 그리는 기능에 대해서는 사각형의 크기가 커질수록 그래픽 가속 영역이 수행되는 비율이 제곱으로 증가하므로 성능 차이가 커진다. 그 밖에 선, 다각형, 윈도우 복사 등의 그리기 기능에 대해서도 대체적으로 가속 모듈을 사용한 경우 성능 향상을 가져온다.

5. 결론 및 추후 연구 과제

본 논문에서는 임베디드 시스템을 위한 그래픽 윈도우 시스템의 그래픽 처리 성능을 향상시키기 위해 그래픽 하드웨어가 제공하는 다양한 2D 그래픽 가속 기능들을 고려한 Tiny X 서버용 그래픽 가속 모듈을 구현하였다.

또한, 성능 평가 프로그램을 통하여 구현된 그래픽 가속 모듈을 사용한 경우가 그렇지 않은 경우에 비해 2D 그래픽 처리 성능이 월등하게 향상된다는 것을 보여주었다.

향후 다양한 임베디드 환경에서 많이 사용되는 이동과 복제 모듈, 멀티미디어 응용을 위한 영상 혼합 모듈, 3D 그래픽 가속 모듈 등에 대한 연구들이 계속 이루어져야 한다.

참고문헌

[1] 김성우, 이경우, "임베디드 그래픽 윈도우 시스템 기술", 정보처리학회지, 2001 년 6 월
[2] Foley, van Dam, Feiner, and Hughes, Computer Graphics: Principles and Practice, 2nd ed., Addison Wesley, 1997
[3] XAA HOWTO, Xfree86 X source 4.2+, xc/programs/Xserver/hw/xfree86/xaa/
[4] http://www.xfree86.org
[5] Susan Angebranndt etc, Definition of the Porting Layer for the X v11 Sample Server, X Window Consortium, DEC.
[6] XFree86 X server "New Design" (DRAFT), http://www.xfree86.org/~dawes/4.3.0/DESIGN.html
[7] Bill Ball, The New Xfree86, Prima Publishing, 2001