리눅스 3D 그래픽 기술 동향

Technical Trend of 3D Graphics for Linux Platforms

임베디드 S/W 기술 동향 및 연구 개발 현황

최승한 (S.H. Choi) 임베디드GUI연구팀 선임연구원
설동명 (D.M. Seol) 임베디드GUI연구팀 선임연구원
안성호 (S.H. Ahn) 임베디드GUI연구팀 선임연구원
이경희 (K.H. Lee) 임베디드GUI연구팀 팀장

목 차

X 서버는 개발된 지 20년 동안 그리 큰 변경이 이루어지지 않았다. 하지만, 그래픽 하드웨어는 사양은 3D 그래픽 고성능으로 변화되었다. X 서버는 2D 그래픽을 처리하기 위한 그래픽 시스템으로 설계되었다. 3D 그래픽 처리를 위해서 OpenGL 라이브러리를 X 서버와 연동하는 작업을 해왔지만 구조적으로 3D 그래픽 드로잉을 처리하는 것은 X 서버이기 때문에 성능에 한계를 드러내게 되었다. 마이크로소프트사 DirectX 구조처럼 직접 그래픽 하드웨어를 제어해서 3D 그래픽 처리를 구현하고 있으며 리눅스 진영은 X 서버와 공존하면서 3D 그래픽 처리를 향상시키기 위한 연구를 진행하고 있다. 본 고에서는 리눅스 진영에서 현재 진행중인 향상된 3D 그래픽 처리에 대한 기술 동향을 설명한다.

Ⅰ. 서론

리눅스 운영체제 환경에서 현재 널리 사용되고 있는 그래픽 시스템으로 X 윈도 시스템이라고 할 수 있다. 1984년 유닉스 개발 공동체에서 진행중이던 Athena 프로젝트의 일환으로 개발된 그래픽 시스템으로 20년 넘게 사용되고 있다. 오픈 소스, 쉬운 이식성 등의 여러 장점을 가지며 리눅스 데스크톱을 개발하는 데 많이 사용되어져 왔다.

X 서버는 태어나서 20년 동안 그리 큰 변경이 이루어지지 않았다. 하지만, 그래픽 하드웨어의 사양은 3D 그래픽 고성능으로 변화되었다. X 서버는 2D 그래픽을 처리하기 위한 그래픽 시스템으로 설계되었다. 3D 그래픽 처리를 위해서 OpenGL 라이브러리를 X 서버와 연동하는 작업을 해왔지만 구조적으로 3D 그래픽 드로잉을 처리하는 것은 X 서버이기 때문에 성능에 한계를 드러내게 되었다. 마이크로소프트사 DirectX 구조처럼 직접 그래픽 하드웨어를 제어해서 3D 그래픽 처리를 구현하고 있으며 리눅스 진영은 X 서버와 공존하면서 3D 그래픽 처리를 향상시키기 위한 연구를 진행하고 있다. 본 고에서는 리눅스 진영에서 현재 진행중인 향상된 3D 그래픽 처리에 대한 기술 동향에 대해 설명한다.

Ⅱ. X 서버환경에서 3D 그래픽을 위한 가속 기술

이 장에서는 리눅스 시스템의 GUI인 X 윈도상에서 하드웨어 가속을 사용하여 3D 그래픽 환경의 속도를 빠르게 하기 위한 방법에 대해서 설명한다.

1. Mesa 3D

리눅스에서 3D 그래픽의 구현이 가능하게끔 하는 수단 중의 하나가 OpenGL이다. OpenGL은 DirectX와 달리 여러 운영체제에서 쉽게 이식할 수 있는 3D 그래픽 라이브러리라는 장점을 가지고 있기 때문이다. 그러나, 대부분의 상용 운영체제에서는 SGI에서 라이선스를 받은 정식 OpenGL 라이브러리를 지원하지만, 리눅스에서는 그렇지 못하였다. 이런 이유로 Mesa 3D라는 이름으로 OpenGL API를 실제로 거의 완벽하게 구현한 공개 라이브러리가 개발되었다.

특히, Mesa 3D에서는 3D 그래픽 가속 기능을 소프트웨어적으로 구현하여, 그래픽 하드웨어가 3D 가속 기능을 지원하지 않더라도 OpenGL 응용의 지원이 가능하다.

아직까지는 상용 OpenGL에 비해 제공하지 않거나 부족한 부분이 없지는 않지만, 일반적으로 사용하는 데에는 거의 무리가 없다. 따라서 OpenGL과 Mesa는 같은 의미로 사용이 가능하다[1].

2. DRI

리눅스에서는 2D 그래픽 처리를 위해 커널 프레임버퍼를 이용하거나 X 윈도 시스템 환경을 사용하는 것이 일반적이다. 이런 상황에서 OpenGL 3D 응용을 위해서는 Mesa 3D 라이브러리를 사용하여야 하지만, 특별히 통합된 구조가 없으면 하드웨어적인 3D 가속 기능을 활용하기 어렵다. 이런 어려움을 해결하기 위해 DRI라는 통합 구조가 필요하게 되었다[2].

즉, DRI는 커널 및 X 서버에서 그래픽 하드웨어의 3D 가속 기능을 직접적으로 제어할 수 있도록 함으로써, X 윈도상에서 OpenGL을 사용할 때 더욱 빠른 3D 가속 수행을 가능하게 한다. 즉, DRI를 사용함으로써 응용 프로그램에서 X 서버로 패킷을 보내지 않고 직접 하드웨어를 제어할 수 있게 된다.

초기에는 소프트웨어 렌더링만을 지원했고, 그 이후에는 하드웨어 가속 기능을 지원하는 방법으로 Utah GLX와 Glide 등을 사용하는 방법이 개발되었다. 그러나 Glide는 부두 계열의 그래픽카드에서만 사용이 가능하였고 Utah GLX는 X 윈도 시스템 중 하나인 XFree86의 4.0 버전 이후에 DRI로 통합되었기 때문에 현재 최선의 하드웨어 가속을 위한 방

법은 DRI라 할 수 있다.

따라서, XFree86에 포함된 DRI 드라이버를 사용하면 클라이언트 프로그램에서 X 서버에 패킷을 보내는 작업이 필요가 없어지게 되면서 직접 하드웨어를 제어할 수 있게 된다.

X 윈도 시스템에서 3D 엔진의 하드웨어 가속을 가능하게 하기 위해서는 DRI가 꼭 필요하다. 만약 DRI를 사용하지 않는다면 3D 엔진은 소프트웨어만으로 렌더링 할 수 밖에 없고 이것은 애플리케이션의 실행 속도에 엄청난 차이를 가지고 올 것이다.

DRI 구조에 대한 자세한 설명은 III장에 기술되어 있다.

3. DRM

DRI를 지원하는 커널 디바이스 드라이버를 DRM이라고 한다. DRM은 그래픽 하드웨어에 대한 동기화된 접근을 제공하고, DRI 보안 정책을 다루며, 일반적인 DMA 엔진 역할을 하여 일련의 그래픽 처리 명령들을 빠르게 수행하도록 도와준다. 또한, DRM은 하드웨어 의존적인 커널 모듈을 통하여 특정한 기능을 확장할 수 있도록 한다[3].

DRM은 다음 세 가지 주된 방법으로 DRI를 지원한다.

(1) DRM은 그래픽 하드웨어에 동시적 접속을 제공한다.

직접 렌더링 시스템은 그래픽 하드웨어에 직접 접근을 위해 경쟁하는 여러 독립체들을 가진다(X 서버, 다중 직접 렌더링 클라이언트, 커널 등). 만약 하나 이상의 요소가 하드웨어에 접근하면 현재 사용중인 하드웨어는 잠긴다.

DRM은 하드웨어 동시 접근을 위해 디바이스 하드웨어 당 단일 락을 제공한다. 하드웨어 락은 X 서버가 2D 렌더링을 실행할 경우, 직접 렌더링 클라이언트가 프레임 버퍼에 읽거나 쓰기가 필요한 fall-back을 실행할 경우, 커널이 DMA 버퍼를 디스패칭할 경우에 요구된다.

(2) DRM은 그래픽 하드웨어로의 접근을 위한 DRI 보안 정책을 시행한다.

X 서버가 root 권한일 경우 일반적으로 /dev/mem/을 이용하여 그래픽 하드웨어상의 MMIO 영역과 프레임 버퍼로의 접근을 획득한다. 그러나 직접 렌더링 클라이언트들은, root 권한으로 접근할 수 없으나 여전히 이와 비슷한 접근이 요구된다. /dev/mem과 같은, DRM 디바이스 인터페이스는 클라이언트가 이런 매핑을 생성하는 것을 허용하지만 다음과 같은 제한이 있다.

- 클라이언트가 현재 X 서버로의 연결설정을 가지고 있다면 클라이언트는 메모리 영역으로의 접근이 허용된다.
- root와 XF86Config 파일(root만이 편집할 수 있는 파일) 안의 특별한 그룹들을 설정하고 접근 권한을 주면 여기에 속한 클라이언트들은 /dev/drm을 이용하여 메모리 영역의 접근이 허용된다. 이것은 시스템 관리자가 믿을 수 있는 사용자 그룹만이 직접 렌더링으로 접근하도록 제한한다.
- 클라이언트는 X 서버가 허용한 메모리 영역으로만 접근하도록 설정한다. X 서버는 이들 영역에 읽기만 되도록 제한한다.

(3) DRM은 일반적인 DMA 엔진을 지원한다.

대부분의 현대 PC-급 그래픽 하드웨어는 FIFO 명령으로 DMA에 접근할 수 있도록 한다. 때때로 DMA 접속은 최적화를 통하여 MMIO 접속보다 좋은 처리량을 제공한다. 이런 카드들을 위해 DRM은 아래와 같은 특징을 가지는 DMA 엔진을 제공한다.

- X 서버는 할당된 메모리 영역에 여러 크기의 버퍼에 대한 다중 풀을 지정할 수 있다.
- 직접 렌더링 클라이언트는 DRM API를 사용하여 이들 버퍼를 가상 어드레스 공간으로 할당한다.
- 직접 렌더링 클라이언트는 DRM으로부터 이들 버퍼들 중 몇 가지를 확보하여, 버퍼를 채우고, 그래픽 하드웨어로 보내기 위해 전송을 요청한다. 작은 버퍼들은 X 서버가 버퍼 디스패치 사이

에 락을 획득할 수 있도록 하기 위해 사용된다.

- DRM은 개개의 OpenGL GLXContext를 위해 DMA 버퍼의 큐를 관리하고, GLXContext 스위치가 필요할 때를 탐지한다. 훅(hook)은 특정 디바이스 드라이버가 커널 안에서 GLXContext 스위치를 실행할 수 있도록 지원한다. DRM은 또한 GLXContext 드래싱(thrashing)을 방지하기 위해 간단한 스케줄링을 수행한다.

4. SDL

SDL은 윈도즈의 DirectX와 비슷하게 리눅스, BSD, MacOS, Win32 등에서 실행되는 게임을 쉽게 개발할 수 있도록 하는 공개 라이브러리이다.

즉, 크로스 플랫폼 멀티미디어 개발용 API로, OpenGL이나, 2D 비디오 프레임버퍼를 통해 비디오뿐만 아니라, 오디오, 키보드, 마우스, 조이스틱, 3D 하드웨어 등을 low level까지 쉽게 제어할 수 있는 라이브러리를 제공한다. 이는 주로 MPEG 플레이백 소프트웨어, 에뮬레이터, 다양한 게임 등에 의해서 이용된다.

다시 말해서, SDL은 Linux, Windows, BeOS, MacOS Classic, MacOS X, FreeBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX 등을 지원하며, 비공식적으로는 Windows CE, AmigaOS, Dreamcast, Atari, NetBSD, AIX, OSF/Tru64, RISC OS, and SymbianOS 등도 지원한다[4].

Ⅲ. DRI

마이크로소프트사의 그래픽 하드웨어를 직접 접근해서 빠른 속도의 3D 게임을 할 수 있도록 하는 DirectX에 대응되는 기술로써 리눅스의 DRI 구조가 개발되어 있다.

DRI 구조는 리눅스 X 윈도 환경에서 OpenGL 라이브러리를 이용해서 3D 응용 프로그램이 직접 그래픽 하드웨어를 제어할 수 있도록 해주는 구조이다. 원래 X 윈도는 구조적으로 하드웨어를 직접 접근하는 방법이 존재하지 않는다. 이유는 X 윈도 개발자들이 X 응용 프로그램 개발자들의 안정된 코딩 기술을 신뢰하기보다는 X 윈도의 구조의 제한을 통해서 X 윈도의 안정성을 확보했기 때문이다. 하지만, 현재의 게임, 멀티미디어드 응용 프로그램, 특히 3D 그래픽을 표현하기 위해서 기존의 X 윈도를 통한 하드웨어 접근방법으로는 만족할 만한 성능을 낼 수 없게 되자 그래픽 하드웨어를 직접 제어할 수 있는 DRI 구조를 개발하게 되었다.

DRI 구조를 개발하게 된 곳은 Precision Insight라는 회사이며 처음 XFree86 4.0에 4개의 그래픽 하드웨어를 지원하며 형태로 통합되어 사용되기 시작했다.

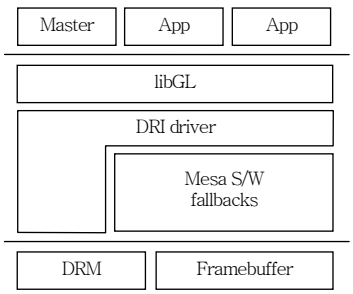
구체적으로 DRI 기술의 목적은 다음과 같다.

- 그래픽 하드웨어 성능 증가
- 다양한 그래픽 하드웨어 설계 지원
- 다수의 응용 프로그램에 동시 렌더링 기능 지원
- 시스템의 의도적인 문제에 대한 보안 기능
- 하드웨어/시스템의 잠김 방지에 대한 신뢰성
- 다양한 OS와 시스템에 쉽게 구현할 수 있는 이식성 제공
- OpenGL과 GLX 스펙과의 호환성
- XFree86 프로젝트와의 통합
- 오픈 소스 정책

(그림 1)은 DRI 구조를 도시한 그림이다.

DRI 구조는 한 개의 모듈이 아닌 크게 4개의 모듈로 구성된다.

첫째, libGL 모듈은 다수의 드라이버 사이에 스위칭 역할, OpenGL API 인터페이스 제공, 그리고,



(그림 1) DRI 구조

GLX 기능을 수행한다. 인다이렉트 렌더링일 경우, libGL 모듈은 GLX 프로토콜 메시지를 만들어서 X 서버에 전달한다. 다이렉트 렌더링일 경우, DRI 드라이버를 통하여 3D 제어 명령어를 전달한다.

둘째, DRI 드라이버 모듈은 3D 그래픽을 위한 그래픽 하드웨어의 제어 기능을 수행한다. OpenGL API 명령어를 그래픽 하드웨어 명령어로 변환하여 다음에 설명할 DRM 모듈을 통하여 그래픽 하드웨어에 전달한다.

셋째, DRM 모듈은 DRI 드라이버 모듈에서 전달된 그래픽 하드웨어 명령어를 그래픽 하드웨어에 전달하는 기능과, 다수의 3D 응용 프로그램이 동시에 그래픽 하드웨어를 접근할 수 있도록 조정하는 기능, 그리고, 그래픽 하드웨어 다운으로부터 응용 프로그램도 다운되는 것을 방지하기 위한 보호 기능을 수행한다.

넷째, Mesa S/W fallbacks 모듈은 그래픽 하드웨어가 특정 3D 기능을 제공하지 못할 경우 소프트웨적으로 처리하기 위한 기능을 수행한다.

지금까지의 설명처럼 DRI 구조는 X 윈도 구조처럼 그래픽 명령어를 서버에 보내서 서버가 처리하는 구조가 아닌, 직접 그래픽 하드웨어에 그래픽 명령어를 전달하는 구조로 데이터 전송의 오버헤드를 줄일 수 있다. 하지만, 그래픽 하드웨어에서 그래픽 콘텍스트 스위칭을 처리해야 하므로 콘텍스트 스위칭을 빠르게 처리할 수 있는 구조의 그래픽 하드웨어가 필요하게 된다. 마이크로소프트에서는 이미 이런 문제의 필요성을 느끼고 DirectX 버전 10을 지원하는 그래픽 하드웨어에 대해서는 콘텍스트 스위치를 빠르게 처리하도록 요구하고 있다.

(그림 2)는 X 서버와 연동되는 DRI 구조의 제어 흐름도를 도시한 그림이다. X 서버와 연동되서 동작하는 이유로 3D 그래픽은 OpenGL과 DRI 구조를 이용해서 표현하지만 기본적인 윈도창과 2D 그래픽은 X 서버를 이용한다.

인다이렉트 3D 그래픽을 처리하기 위한 제어 흐름도의 순서는 “OpenGL 3D 프로그램 → GLX → Mesa S/W 렌더링 코드 → X 서버 2D 드라이버 → 그래픽 하드웨어”가 된다.

DRI 구조에서 지원되는 CPU 구조는 Intel i386 이다.

현재 DRI 구조에서 지원되는 그래픽 하드웨어는 다음과 같다.

- 3dfx, supported on Intel x86, AMD and Alpha

  Voodoo5 5500, Voodoo4 4500, Voodoo3 3500 TV, Voodoo3 3000 AGP, Voodoo3 3000 PCI, Voodoo3 2000 AGP, Voodoo3 2000 PCI, Voodoo Banshee, Velocity 100/200

- Matrox, supported on Intel x86 and AMD: Matrox G200, Matrox G400

- Intel i810/i815/i830 (motherboard chipsets)

  i810, i810-dc100, i810e, i815, i830

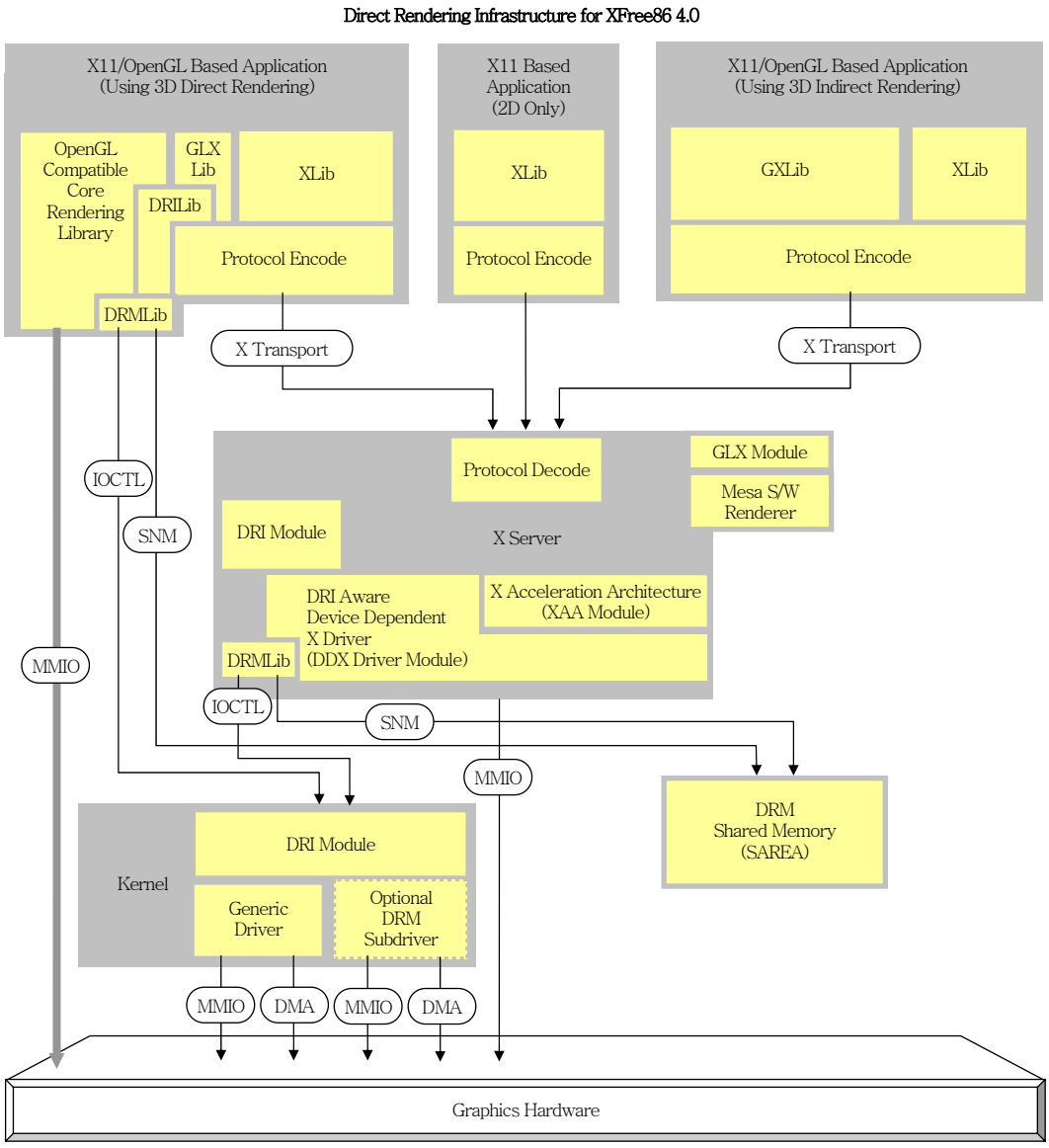
- ATI Rage 128, supported on Intel x86, AMD and Alpha

  Rage Fury, Rage Magnum, XPERT 2000, XPERT 128, XPERT 99, All-in-Wonder 128 Rage 128 PCI (Alpha-based systems)

- ATI Radeon, supported on Intel x86, AMD and Alpha

  Radeon SDR AGP, Radeon DDR AGP, Radeon 32MB SDR PCI (Alpha-based systems)

현재 DRI 프로젝트는 오픈 소스 프로젝트로 진행중에 있으며 누구나 이 프로젝트에 참여해서 개발할 수 있다. DRI 개발의 현재 목표는 높은 성능보다는 안정성에 중점을 두고 개발중에 있다. 즉, 마이크로소프트의 DirectX에 비해서는 안정성 및 성능이 떨어지는 측면이 있지만 개발자들의 많은 참여가 있기 때문에 빠른 시간 내에 극복될 수 있다고 생각된다. 또한, 새로운 그래픽 하드웨어에 대한 드라이버 개발도 병행되어야 한다. OpenGL 라이브러리의 확장에 따른 호환성을 유지하기 위한 개발도 필요하다.



(그림 2) X 서버와 연동되는 DRI

Ⅳ. Xegl

최근 2D 그래픽 애플리케이션들과 윈도 시스템들의 경향은 alpha blending, image transformations, anti-aliasing들과 같은 다양한 그래픽 기능을 요구하고 있다. 이 기능들은 새로운 사용자 인터페이스에 좀더 다양한 효과를 줄 수 있도록 하는 데 필요한 기능들이다[5].

그래픽에서의 다양한 기능들의 지원과 이 기능들의 성능을 최대한 발휘하기 위해서는 그래픽 카드에서의 지원과 가속이 절대적으로 필요하다. 현재 그래픽 카드에서의 기능 지원은 2D 보다는 3D를 위한 다양한 기능들을 지원하고 있으며 이들 기능의 가속도 매우 빠른 속도로 지원하고 있다. 이러한 3D 그래픽 라이브러리를 기존의 2D 그래픽 라이브러리를 이용한 시스템에서 2D 대신에 3D를 이용하는 형태

의 시스템이 부분적으로 개발되고 있고 그 중의 하나가 앞으로 설명할 Xegl이다.

1. Xegl 구성 요소

Xegl은 2005년 8월에 공개되어 현재 진행중인 open source 프로젝트로서 3D 그래픽 라이브러리를 이용하여 X 윈도를 동작할 수 있도록 하는 것으로 기존에 개발되어 있는 다양한 요소들을 사용하고 있다[6].

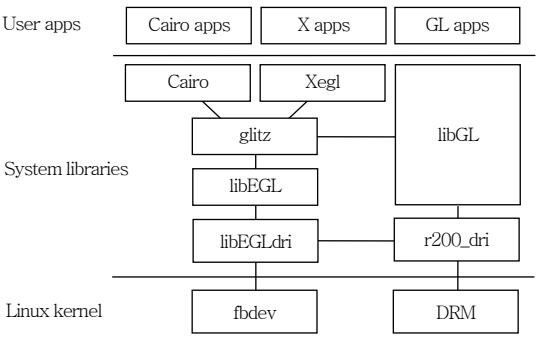
Xgl은 OpenGL을 기반으로 한 X 윈도를 말한다. 기존의 X 윈도에서도 OpenGL을 지원하지만 이 경우 X 윈도의 확장(extension)으로 지원하는 것이고 Xgl은 모든 기능의 OpenGL, 즉 3D 라이브러리를 이용하여 동작하는 것을 말한다.

Xegl은 Xgl의 한 실행 예로써 Mesa/OpenGL을 EGL 확장을 이용하여 동작시키는 것을 말한다. EGL은 Khronos 그룹에서 제안한 구조로서 공용 플랫폼 인터페이스 레이어를 포함하고 있는데, EGL은 플랫폼이나 운영체제와 상관없이 그래픽 작업을 사용할 수 있게 도와주는 것이다. 즉 플랫폼이나 운영체제가 다른 환경에서 작동하고 있는 렌더링 엔진에서도 원하는 그래픽 작업을 할 수 있도록 함수들을 지원해 주는 역할을 한다[7].

다시 말하면 EGL을 기반으로 작성한 프로그램은 하부 윈도 시스템이 리눅스 윈도 시스템 또는 MS의 윈도 시스템 기타 다른 윈도 시스템이든 EGL이 지원되면 작성한 프로그램을 동작시킬 수 있도록 해주는 것을 말한다.

(그림 3)은 Xegl 전체 구조도이다. 리눅스 커널 레벨에서 지원하는 프레임버퍼와 DRM을 이용하여 그래픽 카드를 제어하고 3D 그래픽 라이브러리는 OpenGL의 open source인 Mesa를 이용한다.

OpenGL을 이용하여 그래픽 시스템을 지원하는 Glitz를 이용하여 Cairo와 X 윈도시스템인 Xegl을 시스템 라이브러리 형태로 지원한다. 사용자나 개발자들은 Cairo, X Windows API를 이용하여 기존의 프로그램 개발 방법과 같은 형태로 개발을 하면 하드웨어 가속이 지원되는 3D 그래픽 라이브러리를



(그림 3) Xegl 전체 구조도

이용하여 해당 프로그램을 동작시킬 수가 있다.

Xegl을 구성하는 모듈들을 좀더 자세히 살펴보면 다음과 같다.

• FBdev

FBdev는 그래픽 시스템의 기본 모드로서 모든 그래픽 카드에서 제공하는 표준 기능을 이용하여 그래픽 시스템을 지원하는 모듈이다. 이 모듈은 리눅스의 커널에서 지원하는 부분으로서 Xegl을 사용하기 위하여서는 리눅스 커널 버전이 2.6.13-rc5 이상이 요구된다. 이하의 버전인 경우 패치가 필요하다.

• DRM/Mesa/DRI

DRM/Mesa/DRI는 앞 부분의 DRI 절에서 이미 설명한 부분으로 해당 부분을 참조하면 된다.

• Glitz/Cairo

Glitz는 OpenGL을 이용하여 X render를 구현한 모듈이다. 이 모듈은 그래픽 처리에 있어서 하드웨어 가속 지원하는 기능을 제공하고 있다.

Glitz 모듈은 Cairo에서 3D 그래픽을 지원하기 위하여 구현된 모듈로서 Cairo는 다중 출력을 지원하는 고급의 2D 그래픽 라이브러리이다. 현재 Cairo는 X 윈도 시스템 Win32 윈도 시스템에서 동작이 가능하며 OpenGL, Quartz, XCB, PostScript, PDF file 등을 지원하고 있다.

• Xgl

Xgl은 OpenGL을 이용한 X 서버이다. Xgl 또한

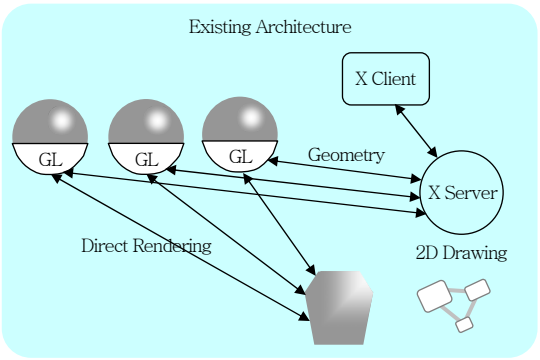
현재 진행중인 프로젝트로서 현재 50~60% 정도 진행되어 있는 상태이다. Xgl의 진행방향 중에 한 방향이 Xegl이다.

2. OpenGL 기반의 X 서버

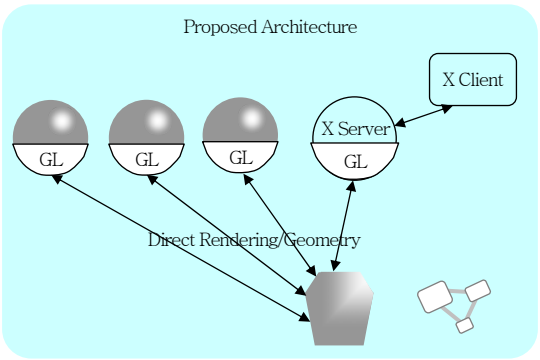
현재의 X 서버는 2D 그래픽을 이용하는 윈도 시스템에서 개별적으로 3D 그래픽 라이브러리를 지원하여 3D 애플리케이션을 동작시키고 있는 형태를 취하고 있다.

2D와 3D를 통합하여 X 윈도도 하나의 GL 애플리케이션으로 하는 방식이 Xgl이다.

(그림 4)는 기존의 X 서버와 3D 애플리케이션과의 관계를 보여주고 있고, (그림 5)는 3D 기반의 X 서버와 다른 애플리케이션과의 관계를 보여주고 있다.



(그림 4) 기존의 X 서버 구조



(그림 5) OpenGL 기반의 X 서버 구조

3. Xegl 현재 진행 상태

Xegl은 현재 Radeon R200 그래픽 카드에서 동작하는 테스트 버전만 공개된 상태로 이 버전도 2개의 그래픽 카드를 이용하여 비디오 바이어스를 수정하여 실행시켜야만 하는 형태로 일반적으로 사용하기는 어려운 형태이다. 이 테스트 버전을 안정화 시키고 일반적으로 사용할 수 있는 상태로 만들기에는 많은 인력과 시간이 필요한 상태이다.

V. 결언

지금까지 리눅스 운영체제를 위해서 현재 진행중인 향상된 3D 그래픽 처리에 대한 기술 동향을 살펴보았다. 현재 사용중인 3D, 멀티미디어 프로그램의 대부분은 하드웨어, 그래픽 라이브러리에 빠른 속도의 3D 그래픽 처리를 요구한다. 또한, 리눅스 데스크톱의 형태도 3D 형태로 개발중인 프로젝트가 진행되고 있다. 요즈음 마이크로소프트와 같은 상용 소프트웨어 진영에 대응하여 리눅스와 같은 오픈 소스 소프트웨어의 필요성이 대두되고 있는데 일반 사용자들이 리눅스를 많이 사용하기 위해서는 다양한 응용 프로그램이 개발되어야 하며 이를 위해서는 응용 프로그램들이 쉽게 개발될 수 있고, 마이크로소프트 윈도 환경에서와 같은 성능을 낼 수 있는 3D 그래픽 기술의 개발이 더욱 이루어져야 한다.

약어 정리

API	Application Programming Interface
DMA	Direct Memory Access
DRI	Direct Rendering Infrastructure
DRM	Direct Rendering Manager
EGL	Embedded Graphic Library
Fbdev	Linux framebuffer console graphics
FIFO	First-In First-Out
GLX	OpenGL Extension to the X window system
MMIO	Memory-Mapped I/O

SDL	Simple DirectMedia Library
SGI	Silicon Graphics, Inc.
Xegl	Xgl on Mesa/OpenGL Extensions
Xgl	OpenGL based Xserver

참 고 문 헌

[1] Mesa Project, http://www.mesa3d.org/
[2] DRI Project, http://dri.freedesktop.org/
[3] DRM, http://dri.freedesktop.org/wiki/DRM
[4] SDL Project, http://www.libsdl.org/index.php
[5] Peter Nilsson and David Reveman, "Glitz: Hardware Accelerated Image Compositing Using OpenGL," USENIX 2004, pp.29-40.
[6] JonSmirl, Xegl, http://www.freedesktop.org/wiki/Xegl, Aug. 8, 2005.
[7] EGL, http://www.khronos.org/egl/