VRender3D 프로세서를 위한 Mobile 3D Engine 포팅

정일동 ° Alexander O. Fedorov 김용태 이군석

LG 전자

{idjung96, afedorov, ytkim, koonseoklee}@lge.com

Porting Mobile 3D Engine to VRender3D Processor

Il-dong Jung°, Alexander O. Fedorov, Yong-tae Kim, Koon-seok Lee

LG Electronics

요 약

PDA와 같은 이동단말에서 화려한 3D 그래픽을 보여주는 것은 그래픽 시스템 리소스의 제약이 있다. Mobile 3D 엔진은 모델의 움직임을 계산하여 동적으로 3D 그래픽을 만들어 내기 때문에 그래픽 시스템 뿐만 아니라, 충분한 성능의 프로세서와 여유의 메모리까지 지원되어야 한다. 본 논문에서는 Mobile 3D 엔진의 제약 사항과 그 해결 방법을 제시하였다. ARM9 Core를 기반으로 3D 가속 기능을 가진 VRender3D에 실제로 OpenGL/ES를 기반으로 하는 Mobile 3D 엔진을 포팅 (porting) 하고, 그 성능을 동적인 3D 영상으로 평가하였다.

1. 서 론

최근 Mobile 3D 기술이 화두로 떠오르면서 Mobile Contents가 2D기반에서 3D기반으로 이동하고 있다. 초창기에는 실제 3D기반 Contents가 아니라, 3D로 제작된 이미지를 보여주었다. 사용자들은 같은 3D 이미지를 계속 보여주는 것보다는, 입력에 따라 새로운 모습을 보여주는 Contents를 기대하게 되었다.

사용자의 입력에 따라 실시간으로 변하는 3D를 보여주기 위해서는 PC 게임이나 솔루션에서 사용하던 3D 엔진이 필요하다. 3D 엔진은 3D 모델링 데이터와 애니메이션 데이터를 조합하여 실시간으로 렌더링 (Rendering) 하여 화면에 3D로 표시해 준다. 이 작업은 계산량일 필요하고 저장 공간을 소모하며, 많은 양의 데이터 이동을 수반한다.

따라서 프로세서의 속도와 저장 공간의 크기와 속도, 화면 크기 등을 고려하여 Mobile 환경에 적합한 3D 엔진이 필요하다. 품질이 좋은 모델을 실시간으로 표현하기 위해서는, 수 천 개의 점과 선으로 이루어진 모델에 텍스처를 입히고 광원을 처리하는 연산을 고속으로 처리하기 위한 3D 가속 기능도 필수이다.

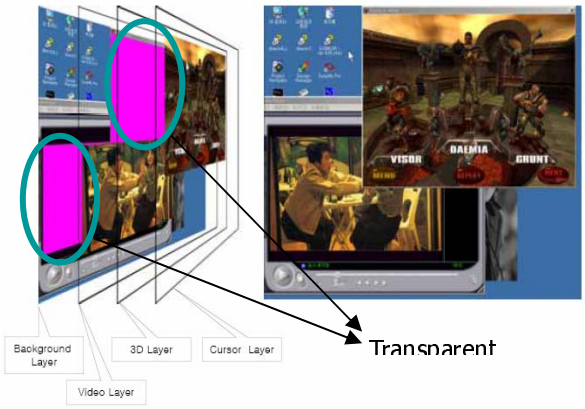
본 논문에서는 ARM9 Core를 기반으로 3D 가속 기능을 가진 VRender3D에 Mobile 3D 엔진을 제약 사항을 회피하여 포팅 (porting) 하고, 그 성능을 3D 영상으로 평가하였다.

2. VRender3D 프로세서

VRender3D는 3D 가속 기능을 가지고 있는 ARM920T Core 기반의 모바일 프로세서이며 최대 동작 주파수는 200Mhz이다. 게다가 OpenGL/ES기반의 3D가속 기능을 지원하는데 다음과 같은 특징을 지닌다.

- 3D Graphics GTE (Geometric Transform Engine)
- 4 Floating Vector Engines. (24bit)
  - -Floating Point Adder
  - -Floating Point Multiplier
  - -Floating Point Divider
  - -Floating Point Reciprocal
- World / Camera Transform
- Lighting
  - -Ambient Light
  - -Point Light
  - -Directional Light
- 3D Graphics Rasterizer
- Perspective Corrected Multi Texture Mapping
  - -Color Format : 4 / 8 / 16 Formats
- Support Enhanced Z-Buffer
- Alpha Blending compatible with Direct3D
- Hardware Dithering

VRender3D는 윈도우 내부에서 3D 화면을 보여주기 위해서는 3D 영역을 따로 설정해야 한다. [그림1]은 실제 윈도우 화면 3D영역, 동영상 영역이 어떻게 구성되는가를 보여 준다. VRender3D에서 출력하는 화면 크게 4개의 층으로 구성되어 있는데, 제일 아래는 윈도우, 그 위에는 동영상 영역과 3D영역, 그리고 제일 위에는 화면에 보이는 커서를 출력하는 영역이다.



[그림 1] VRender3D의 화면 구성

윈도우 내에 3D 화면을 출력하려면 윈도우 내에 투명색 (윈도우의 색테이블에서 MAGENTA값) 처리를 하여야 한다. 이는 VRender3D의 윈도우 라이브러리의 메커니즘이므로 반드시 지켜야 한다.

```
/* 시스템 초기화 */
MES_Initialize();
MES_CreateLayer_3D(m_Width, m_Height);
MES_ShowLayer_3D(0, 0, m_Width, m_Height);
 ¡ ¦
/* 시스템 해제 */
MES_HideLayer_3D();
MES_Deinitialize();

/* 3D 영역 시각화 */
CBrush br, *pOldBr;
CRect rt(0,0,480,800);

br.CreateSolidBrush(RGB(255,0,255));

pOldBr = (CBrush*)pDC->SelectObject(&br);
pDC->FillRect(&rt, &br);
pDC->SelectObject(pOldBr);
```

[그림 2] 3D 영역을 초기화하는 코드

3. Mobile 3D 엔진

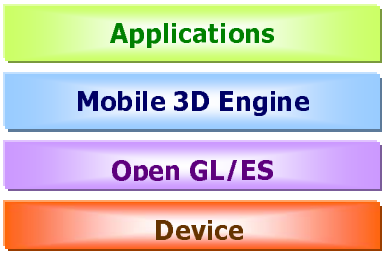
Mobile 3D 엔진은 PDA, 휴대폰 등과 같이 휴대용 기기 상에서 실시간 3D 그래픽 렌더링을 구현해주는 엔진을 의미하며, Notebook PC에서의 mobile 3D와는 구별되며 각종 3D 아바타 및 게임, 광고 등과 같은 서비스를 구현 가능케 한다. Mobile 3D 게임 엔진의 종류는 3D 게임을 이동통신 단말기 상에서 구현시켜 주는 기술에서부터 대용량의 3D이미지를 이동통신 단말기에서 볼 수 있도록 압축 및 복원을 지원하는 3D 이미지 인코딩 기술까지 다양하다[1].

Mobile환경에서 부족한 자원 상황에도 불구하고 Mobile 3D 엔진은 다음과 같은 주요 기능을 제공한다.

① 3D 모델과 모션 데이터의 조합[5]

② 모션 데이터와 모션 데이터의 조합

Mobile 3D 엔진을 이용한 응용 프로그램은 [그림3]로 구성된다. 목적에 맞게 Mobile 3D 엔진의 기능을 선택해서 조합하여 사용한다.



[그림 3] Mobile 3D 엔진의 구성도

OpenGL/ES는 Khronos Group에서 개발 중인 Mobile 3D API이다. OpenGL/ES는 OpenGL을 기반으로 하는 소프트웨어와 하드웨어 사이의 가벼운 인터페이스를 제공하는 저수준 API이며, 기본적으로 OpenGL 1.3 파이프라인 Feature를 사용하며, 다음을 포함한다. [2].

① Geometry Processing
② Rasterization
③ Texture Mapping
④ Fragment Processing
⑤ Frame buffer

Mobile 3D 엔진의 특징 및 요구사항은 다음과 같다

① 물리적 크기

3D 렌더링에서 필수적으로 Frame buffer, Depth buffer, Texture memory의 최소한 3가지 종류의 메모리를 필요로 한다. 하지만 대부분의 이동단말에서는 주메모리조차도 부족한 상황이다. 따라서 이들을 주메모리에 올리기는 어려우며, 주메모리로 올린 후 원하는 속도로 액세스하기도 어렵다. 따라서 3D를 위한 별도의 메모리가 존재해야 하며, 제한된 물리적 크기

내에서 구현하기 어렵다[3].

② 작은 화면 해상도

PDA, 핸드폰과 같은 이동 단말은 작은 화면 해상도를 갖고 현재 출시된 PDA는 640*480이하의 화면을 가진다. 따라서 렌더링시 필요한 Pixel의 수가 PC보다 작으며, 이로 인해 계산량이 작기 때문에 이동 단말에서도 충분한 가능성을 가지고 있다.

③ 데이터 압축 기술 활용

3D는 많은 좌표 데이터와 텍스쳐 데이터 처리를 위하여 3D의 경우보다 많은 저장 메모리를 사용해야 되기 때문에, 개발하는 입장에서는 상당히 답답한 메모리 상황이다. 이 문제를 해결하려면 데이터 압축 기술을 최대한 활용하여 저장 메모리를 줄이고 최근 많이 사용하고 있는 확장 메모리카드를 활용하는 방안을 강구해야 한다[1].

④ FPS (Frame per second)

제작하는 응용 프로그램이 어느 정도의 FPS를 요구하느냐에 따라 결정된다. 정적인 게임의 경우에는 낮은 프레임을 선택하고 많은 폴리곤을 사용해서 화려한 화면을 만들어야 하고, 반대로 동적인 게임의 경우에는 동적인 화면을 만들기 위해서 폴리곤 수가 적은 모델을 사용하여 높은 프레임을 만들어야 한다.

⑤ 충돌 검출

충돌 검출 기능은 두 물체가 3D 공간에서 충돌하였는지를 검사하는 것이다. Mobile 3D 엔진을 활용해서 게임을 만들 때 중요한 기능이다. 충돌 검출은 실시간 물리학 엔진의 중요한 부분들 중 하나이다. 현재 사용되는 충돌 검출법은 vertex와 vertex, vertex와 edge, vertex와 face, face와 edge, sphere와 edge, sphere와 face, box와 edge, box와 face, box와 box, sphere와 sphere 등이 있다[4].

4. Mobile 3D 엔진 포팅

OpenGL/ES 를 기반으로 하는 Mobile 3D 엔진은 특별한 수정없이 다른 플랫폼으로 이식할 수 있어야 한다. 하지만, VRender3D의 OpenGL/ES는 API가 일치하지 않아서 부득이하게 Mobile 3D엔진을 수정하여야만 한다.

① Floating point 기반 Vertex

Mobile 3D엔진은 Vertex의 데이터 타입은 보통 4바이트의 int타입을 정수부 2바이트, 소수부 2바이트를 사용하는 고정 소수점을 많이 사용한다. 하지만 VRender3D는 Floating point기반의 vertex를 사용하기 때문에 입력하고 처리하기 위해서는 데이터 변환 과정을 거쳐야 한다. 데이터 변환을 위해서 200Mhz의 VRender3D에서 계산과 메모리 대역을 사용해야 하기 때문에 다른 성능을 떨어뜨리는 것을 피할 수가 없었다.

② Texture 크기

VRender3D는 메모리 관리를 위해서 3D에서 사용하는 Texture를 512*512의 크기로 제한을 두었다. 이 크기 이상의 Texture를 사용하는 경우가 있을 수 있는데, 다음의 2가지로 생각해 볼 수 있다.

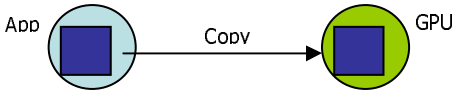
첫째, 3D모델의 Texture는 제한된 크기를 넘지 않아야 한다. 인간 혹은 그와 비슷한 3D 모델은 머리와 몸통, 팔, 다리, 손 등의 각 부위를 다른 객체로 구성하고 Texture를 사용하기 때문에 실제로 제한된 크기를 넘는 경우가 드물 것이다. 그 이상의 Texture는 사용하지 않도록 한다.

둘째, 배경은 보통 2차원 평면을 3D 모델 뒤에 배치하여 만들어 낸다. 따라서 배경에서 사용하는 Texture는 디스플레이의 크기에 의존한다. 320*240 해상도를 가진 4인치 디스플레이를 쓰는 경우는 문제가 되지 않지만, 640*480의 6.4인치 혹은 800*480의 7인치 디스플레이를 사용하는 경우에는 배경 그림을 표현할 수가 없었다. 그래서 배경 그림을 512*480, 128*480의 Texture의 조합 혹은 512*480, 288*480의 Texture의 조합을 사용해서 배경 그림을 만들어야 한다.

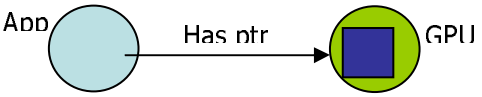
③ Vertex Buffer

기존에는 [그림4.a]와 같이, 화면을 그릴 때마다 응용 프로그램에서 관리하는 데이터를 GPU의 메모리에 복사하였다. 하지만 3D 모델을 초당 10~20회를 수정하기 때문에 필요한 FPS 만큼 복사를 해야 한다. 메모리 및 CPU의 사이클을 사용하므로 속도 Vertex buffer를 수정할 수 있는 횟수가 줄어든다.

그래서 [그림4.b]처럼 GPU의 메모리의 포인터를 사용해서 응용 프로그램이 직접 데이터를 수정하도록 하였다. 메모리와 CPU의 사이클을 절약할 수 있기 때문에 FPS를 증가시킬 수 있다.



[그림 4.a] 수정 전의 GPU 메모리 관리



[그림 4.b] 수정 이후의 GPU 메모리 관리

5. 성능 평가

포팅된 3D엔진의 성능을 평가하기 위하여 다음과 같은 환경에서 [그림5]의 모델을 사용하였다.

① 화면 크기: 800*480
② 색상 수: 26만 컬러(18 bit)
③ 모델의 polygon: 약 4000 개
④ 모델의 내부 노드: 약 30개

[그림 5] 사용한 3D 모델의 그림

응용 프로그램에서 화면을 그리는 루틴에서 초당 호출되는 횟수를 측정하여 13~14FPS의 성능을 보임을 알 수 있었다.

3D 가속기가 초당 100만 폴리곤 (polygon) 을 그려낼 수 있다고 하더라도 메모리의 대역폭과 CPU의 속도의 한계로 인해서 100만 폴리곤을 모두 그려내지 못한다. VRender3D를 기반으로 한 Mobile 3D엔진도 실제 칩 제작사에서 제시하는 70만 폴리곤을 그려내지 못하였다.

6. 결론

다양한 3D 콘텐츠를 사용자에게 제공하기 위해서3D 엔진을 활용하여야 한다. 3D 엔진이 다양하게 활용되기 위해서는 Mobile 환경에서 실행할 수 있어야 한다. 따라서 Mobile 환경의 제약 사항인 계싼량에 의존하는 배터리 사용량와 화면의 크기, CPU의 속도, 메모리 대역폭 등을 극복하여야 한다.

본 논문에서는 Mobile 3D엔진을 VRender3D의 제약 사항을 고려하여 포팅하였다. 포팅한 Mobile 3D엔진을 활용한 응용 프로그램에서 그 성능을 확인하여 4000폴리곤을 활용하는 3D 모델을 13~14FPS의 성능을 보였다.

Mobile 3D 엔진을 포팅만 하면 그 작업은 무의미한 것이다. 새로운 플랫폼에 포팅한 엔진이 최대한 활용할 수 있는 양한 게임을 포함한 응용 프로그램을 개발해야 한다. 그리고 향후에는 Mobile 3D엔진을 다른 환경에도 포팅하고 엔진의 성능을 향상할 필요가 있다.

7. 참고

[1] 한국게임산업개발원, 2002, 대한민국 게임 백서, 2002
[2] OpenGL/ES, Khronos, http://www.khronos.org, 2003
[3] Ju-ho Sohn외 2명, “Optimization of 3Portable System Architecture for Real-time 3D Graphics”, ISCAS2002, 2002
[4] 강윤미, 박용범, “3D 게임에서 충돌 검출 모델의 효율성 분석”, 20회 한국정보처리학회 추계학술발표대회, 2003
[5] 고미드 홈페이지. ,http://www.gomid.co.kr