Android Emulator의 OpenGL 연산 효율 개선에 관한 연구

김정웅*, 이동렬**, 양해술*
*호서벤처전문대학원 컴퓨터응용기술학과
**중부대학교 정보통신학과
e-mail:jwk@korea.com

A Study on Improvement for OpenGL Execution Efficiency of Android Emulator(QEMU)

Jeong Woong Kim*, Dong Real Lee**, Hae-Sool Yang*
*Dept. Application of Computer Technology,
Hoseo Graduate School of Venture
**Dept of Information and Communication Engineering,
Joongbu University

요 약

Android OpenGL ES Issue Report에서 제기된 Android Emulator에서의 OpenGL 연산속도 문제를 Matrix 연산이 많이 사용되는 OpenGL 3D 구현에서 확인하고 이를 개선할 수 있는 방법을 제시한다. Android Emulator에 포함된 OpenGL ES는 소프트웨어 방식의 OpenGL ES 1.5가 사용되고 있다. 이때 Floating Point가 개선되면 3D의 연산 속도를 높일 수 있을 것이다. 이를 위하여 본 논문에서는 Android Emulator를 수정하여 개선하고, 샘플코드를 통해 테스트 결과를 제시한다.

1. 서론

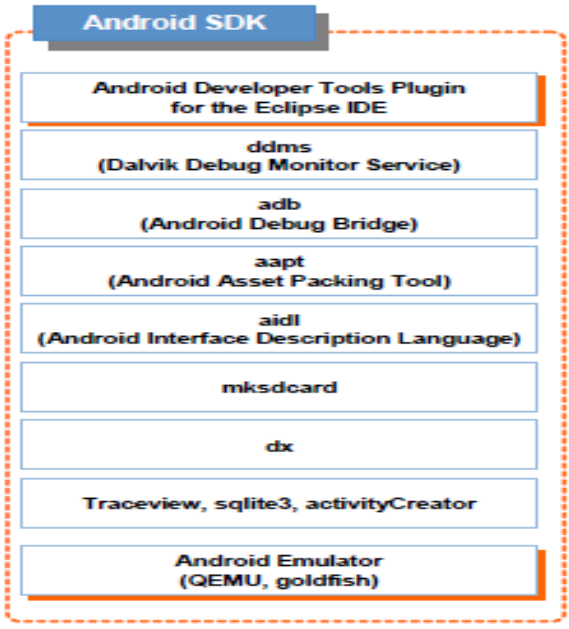
무료 모바일 운용체계(OS)전략에 기반 한 구글(Google)의 모바일 서비스 전략이 개방형 휴대폰 동맹(OHA : Open Handset Alliance)을 통해 웹 접속서비스 주도권을 PC에서 모바일로 넘기면서 웹2.0시대의 공유·개방을 실험하는 시도가 이루어지고 있다. 현재의 웹 운영처럼 수익은 광고를 통해 해결하고 게임·소프트웨어 등의 모바일 서비스 대부분을 무료로 제공할 예정이다. 이는 기존의 스마트폰 OS 시장 진영뿐만 아니라 많은 사람들에게 관심을 갖게 하고 있다[1].

최근 공학적 측면에서도 구글 안드로이드 플랫폼 소스를 이용한 다양한 응용프로그램 개발과 구글 모바일 플랫폼 안드로이드(Android)의 SDK 공개와 오픈 소스 자바 SE 구현 프로젝트인 아파치 하모니, 그리고 JVM과는 다른 형태의 안드로이드 플랫폼의 Dalvik VM에 대한 연구가 활발하게 이루어지고 있다.

본 논문에서는 Android Run-Time에 대한 다양한 이슈들 중에서 Android Emulator에서의 OpenGL 연산속도 문제를 Android Emulator(QEMU) 수정을 통하여 해결하고자 한다.

2. Android

Android는 운영체제, 미들웨어 그리고 핵심 애플리케이션들을 포함하고 있는 모바일 디바이스를 위한 “하나의 소프트웨어 스택”이다. Android SDK의 초기 형태는 Java 프로그래밍 언어를 사용하여 안드로이드 플랫폼 상에서 애플리케이션 개발을 시작하기 위해 필요한 도구들과 API들을 제공하고 있다.



(그림 1) Android SDK[2]

모든 안드로이드 애플리케이션은 Dalvik 가상 머신내의 자신의 인스턴스를 가지고, 자신의 프로세스 내에서 동작한다. Dalvik은 하나의 디바이스가 복수의 VM들을 효

과적으로 실행하도록 만들어졌다. Dalvik VM 최소 메모리 사용하도록 최적화된 Dalvik Executable(.dex) 포맷의 파일들을 실행한다. Dalvik VM은 레지스터 기반이며, Java 언어 컴파일러에 의해 컴파일된 클래스를 "dx"라는 도구에 의해 .dex 포맷으로 변환된 클래스를 실행한다. Dalvik VM은 쓰레딩과 저수준 메모리 관리와 같은 기능을 위해 리눅스 커널에 기초한다[3][4].

Android Run-Time에 대한 다양한 Dalvik VM 이슈들이 제기되고 있다. Google의 Dalvik VM Source Open과 관련 추후 호환성 또는 발생될 다양한 문제들에 대한 접근, 모바일 플랫폼의 안정성과 오픈 소스 요구에 대한 해법, Initiative의 유지 가능 여부 등이 있다. 또한 특화된 Dalvik VM에 따라 Application의 Performance 시 VM의 역할, Embedded Application에서의 Moore 법칙 유효 여부, VM 내부 구조(Stack-base 또는 Register-base)가 App. 성능에 미치는 영향, JIT Compilation가 App. 성능에 미치는 영향 등이 있다[5].

3. Dalvik VM

안드로이드에 사용되는 Dalvik VM은 레지스터 기반으로 스택 기반의 자바와 형태는 다르지만 실제 명령은 자바와 거의 일대일로 대응되는 구조다. Lua의 예에서도 볼 수 있지만 레지스터 기반 VM은 컴파일러가 코드를 생성하기가 상대적으로 어렵고 명령어 자체도 해석기 형태로 구현하면 디코딩하는 부담이 있다. 그러나 스택을 매개로 데이터를 교환하는 것보다 수행해야 할 명령어 수가 줄고 효율적인 구현이 가능하기도 하다. 사실 Dalvik은 기술적인 신규성 보다는 썬의 지적재산권을 우회하고 메모리 제약이 있는 임베디드 시스템에 더 적합한 구조를 선택한 결과라고 생각된다.

안드로이드 SDK에서는 자바 5.0 이상에서 지원하는 Annotation과 Generic을 지원하지 않는 등 특화된 Dalvik VM이 byte-code 단위에서의 platform-independency를 보장해준다는 자바의 특성과는 배치된다. 다양한 환경에서의 VM의 특성과 효용 가치를 분석하고, 최신 자바 버전의 라이브러리들을 안드로이드 플랫폼에서 응용 가능한 기법에 대한 연구가 필요하다.

4. Android Emulator(QEMU) Floating Point

Android OpenGL ES Issue Report에 보면 Android Emulator에서 OpenGL 연산속도가 문제가 되고 있다. Android Emulator에 포함된 OpenGL ES는 소프트웨어 방식의 OpenGL ES 1.5가 사용되고 있다. 그러므로 OpenGL 3D를 구현하기 위해서는 Matrix 연산이 많이 사용되는데, Floating Point가 개선되면 3D의 연산 속도를 높일 수 있을 것이다.

Android Emulator 수정하여 Floating Point 연산 속도를 개선하기 위해서 다음과 같은 과정이 요구된다. 첫째, Android Emulator의 fpu 소스코드에서는 softfloat.c를 사용하고 있다. 속도를 개선하기 위해서는 softfloat-native.c를 아래 내용과 같이 패치 하여 android-emulator-1.0_r2\android-emulator-20081210\qemu\fpu의 파일들을 대체한다.

- 삽입

```
void set_float_exception_flags
        (int val STATUS_PARAM)
int get_float_exception_flags(float_status *status)
float32 uint32_to_float32
        (unsigned int v STATUS_PARAM)
float64 uint32_to_float64
        (unsigned int v STATUS_PARAM)
unsigned int float32_to_uint32
        (float32 a STATUS_PARAM)
unsigned int float32_to_uint32_round_to_zero
        (float32 a STATUS_PARAM)
unsigned int float64_to_uint32
        (float64 a STATUS_PARAM)
unsigned int float64_to_uint32_round_to_zero
        (float64 a STATUS_PARAM)
uint64_t float64_to_uint64
        (float64 a STATUS_PARAM)
uint64_t float64_to_uint64_round_to_zero
        (float64 a STATUS_PARAM)
```

- 삭제

```
float64 float64_trunc_to_int
        (float64 a STATUS_PARAM)
int float64_is_nan(float64 a1)
```

- int를 char형으로 자료형 변경

```
float64_compare
float64_compare_quiet
float64_is_signaling_nan
```

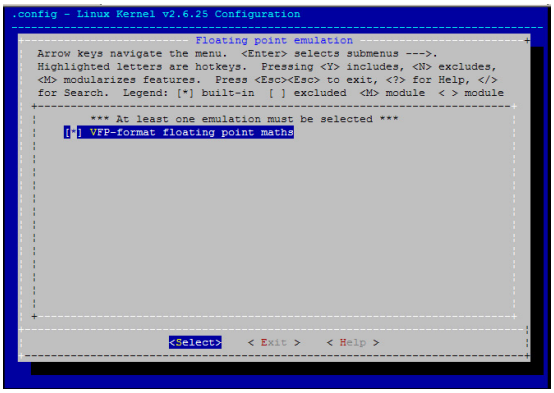
그리고 android-emulator-1.0_r2\android-emulator-20081210\qemu\android\config\config.h 파일에서 아래와 같이 CONFIG_SOFTFLOAT을 undefine 한다.

```
//#define CONFIG_SOFTFLOAT 1
#undef CONFIG_SOFTFLOAT
```

또한, android-emulator-1.0_r2\android-emulator-20081210\qemu\Makefile.android의 softfloat.c 파일을 아래와 같이 softfloat-native.c로 수정한다.

```
line 170: FPU_SOURCES := softfloat-native.c
line 250: FPU_SOURCES := softfloat-native.c
```

android-emulator-1.0_r2\android-emulator-20081210\qemu에서 make 하면, objs 디렉토리에 emulator.exe가 생성된다.



(그림 2) Android_Kernel_Config

둘째, linux kernel config(.config) 파일에서, CONFIG_VFP=y로 설정되어 있어야 한다. 다행이 Android에서 사용하는 linux kernel에서는 VFP 를 사용하도록 설정되어 있다.

셋째, linux 어플리케이션 프로그램을 컴파일 할 때 컴파일 옵션에, "-mfpu=vfp -mfloat-abi=softfp"를 포함해서 컴파일 한다. 즉, Dalvik VM 이나 OpenGL ES를 사용하는 라이브러리들이 이 옵션으로 컴파일 되어야 한다.

5. 연산 속도 테스트

테스트에 사용한 샘플코드는 간단하게 float 변수 사칙연산을 각각 1,000,000번 수행한 시간을 측정한다. 테스트에 사용한 컴퓨터 사양은 CPU : Intel Core2 6400 2.13GHz, RAM: 2G, OS: Windows XP SP3를 사용하였고, 컴파일은 Codesourcery G++ Lite 208q1-126 for ARM GNU/Linux를 사용하였다. 또한, Android Emulator는 libc를 사용하지 않아 라이브러리를 static link 하였고, float_vfp는 "-mfpu=vfp -mfloat-abi=softfp" 옵션을 포함해서 컴파일하고 float_nofp는 옵션 없이 컴파일 한 실행파일이다.

<표 1> 수정하지 않은 Android Emulator 테스트 결과

operation / 0s, us	add	sub	mul	div
elipse time(./float_nofp)	135414	202468	116733	272469
elipse time(./float_vfp)	100574	15769	97731	110618

<표 2> softfloat-native를 적용한 Android Emulator 테스트 결과

operation / 0s, us	add	sub	mul	div
elipse time(./float_nofp)	137863	201425	118687	271949
elipse time(./float_vfp)	68719	67290	66261	78777

위와 같이 vfp 옵션을 주고 컴파일 한 경우에는 거의 2배 정도 성능개선 효과가 있는 것으로 보이고 Android Emulator 에 softfloat-native를 적용하면 약30% 정도의 성능이 개선되었다.

6. 결론

Matrix 연산이 많이 사용되는 OpenGL 3D를 구글 안드로이드 모바일 플랫폼 소스를 이용하여 구현하였을 때 Android Emulator에서의 OpenGL 연산속도가 문제가 된다. Android Emulator에 포함된 OpenGL ES는 소프트웨어 방식의 OpenGL ES 1.5가 사용되고 있다. 이때 Floating Point의 개선을 위하여 Android Emulator의 기존 소스코드를 를 수정하여 버그를 해결하고 패치하여 재구성하였다. float 변수 사칙연산을 각각 1,000,000번 수행하는 샘플코드를 통한 테스트 결과 연산속도가 개선됨을 확인하였다.

참고문헌

[1] http://www.openhandsetalliance.com/index.html
[2] http://www.kandroid.org/board/kandroid_home.php
[3] http://www.android.com/
[4] Professionial Android Application Development (Paperback), Reto Meier| John Wiley & Sons Inc, 2008
[5] Google Android and the Wireless Ecosystem: Will the Mobile Future be Google's Future?, Mind Commerce Publishing, 2008
[6] The Busy Coder's Guide to Android Development ,Commonsware, LLC, 2008
[7] 리눅스 기반의 휴대단말 운영체제 동향 분석, 오승희, 한국전자통신연구원, 2008
[8] 구글의 모바일 폰 서비스 시장 진출과 그 의미, 이양환, 한국방송영상산업진흥원, 2007