멀티미디어 프로세서를 이용한 OpenVG 및 SVG Tiny의 가속

이 환용 백 낙훈*
㈜휴원 경북대학교 컴퓨터학부
hylee@hul.com oceancru@gmail.com

Accelerating OpenVG and SVG Tiny with Multimedia Processors

Hwanyong Lee Nakhoon Baek*
Huone, Inc. Kyungpook National University

요 약

스마트 폰을 비롯한 다양한 임베디드 환경에서 널리 쓰이는 2D 벡터 그래픽스 기술에는 OpenVG와 SVG Tiny가 대표적이다. 고해상도 화면에서 높은 재생속도의 벡터 그래픽스 응용을 위해서는, 이들을 효과적으로 가속해야 한다. 현재까지 OpenVG와 SVG Tiny의 구현 방법에는, 전용 그래픽스 칩과 같은 하드웨어로 구현하는 방법과, 전체를 소프트웨어로 구현하는 방식이 있었다. 현재 시장에서 사용 가능한 벡터 그래픽스 전용 칩들은 상대적으로 고가에 큰 전력을 소비한다. 반면, 소프트웨어 구현에서는 100%에 가까운 CPU 사용률에서도 상대적으로 낮은 성능을 보이고, 이 경우에, 다른 멀티-쓰레드 응용 프로그램들을 방해할 가능성이 컸다. 본 논문에서는, 현재 미디어 재생 기기들과 휴대폰들에 광범위하게 장착되어 있는 상용 멀티미디어용 하드웨어들을 사용하여 OpenVG와 SVG Tiny 양쪽 모두를 가속하는 비용대비 효과적인 방법을 제시한다. 멀티미디어 프로세서들을 효과적으로 사용함으로써, CPU 사용률과 전력 소모량을 줄이면서도 OpenVG와 SVG Tiny를 최소 3.5배에서 최대 30배까지 성공적으로 가속할 수 있었다.

Abstract

OpenVG and SVG Tiny are the most widely used 2D vector graphics technologies for outputs in the various embedded environments including smart phones. Especially, to show high refresh rates on the high resolution screens, it is necessary to effectively accelerate them. Until now, OpenVG and SVG Tiny are available as hardware implementations such as the fully-dedicated graphics chips or full software implementations. Currently available vector graphics silicon chips are relatively expensive and require high power consumption. In contrast, previous full software implementations show lower performance even with almost 100% CPU usages, which would disrupt other multi-threaded applications. In this paper, we present a cost-effective way of accelerating both of OpenVG and SVG Tiny, based on the multimedia-processing hardware, which is wide-spread on the media devices and mobile phones. Through the effective use of these multimedia processors, we successfully accelerated OpenVG and SVG Tiny at least 3.5 times to at most 30 times, even with lower power consumption and lower CPU usage.

* 교신저자 (oceancru@gmail.com)

투고일: 2010-03-12 / 심사일: 2차 2010-04-28 / 게재확정일: 2010-05-11

키워드: OpenVG, SVG Tiny, 멀티미디어 프로세서, 구현
Keywords: OpenVG, SVG Tiny, multimedia processor, implementation

1. 서론

벡터 그래픽스 기능은 화면 크기 및 해상도에 무관한 품질, 상대적으로 작은 파일 크기, 이미지에 대한 무손실 압축 가능성, 확대에 따른 이미지 손상을 피할 수 있다는 등의 다양한 장점을 제공한다[1,2]. 현재 이러한 기능들에 대한 요구는 상당하며, 특히 그래픽스 사용자 인터페이스가 필수적인 스마트 폰을 비롯한 휴대용 임베디드 시스템을 중심으로 꾸준히 증가하는 추세이다. 예를 들어, WAP [3], MMS [4], DCD [5] 등의 Open Mobile Alliance (OMA) 에서 제정한 표준들, JSR226 [6], JSR287 [7], JSR 271 [8] 등의 Java 표준들, MPEG4 part 20: LASeR [9] 등의 ISO/IEC JTC1 SC29 표준들, W3C 의 HTML5 [10], EPUB [11] 등의 *e-book* 표준들에서 벡터 그래픽스 기반의 렌더링을 필요로 한다. 최근에는 애플 아이폰, 구글 안드로이드 등의 스마트폰, 스마트 TV, 셋탑 박스 등에서 다양한 GUI 와 어플리케이션 프로그램들을 제공하기 위해서 벡터 그래픽스 기능을 요구하고 있다.

현재 벡터 그래픽스 기능은 Flash [12], SVG [1,2], Cairo [13], Silverlight [14], PDF, PostScript [15] 등의 다양한 API 와 파일 포맷들에서 사용 가능하지만, 휴대용 기기들에서는 크로노스 그룹에서 제정한 OpenVG 가 현재 가장 적합한 API 로 평가되고 있다. OpenVG 는 로열티를 지급할 필요가 없으며, 다양한 플랫폼에서 순수 소프트웨어 또는 하드웨어 가속을 통하여 사용하기 쉽게 설계된, 낮은 계층의 API 표준이다.

더 크고 해상도가 높은 화면에서도 역동적인 사용자 인터페이스와 풍부한 미디어 서비스를 지원하기 위해서, 벡터 그래픽스 기능에 대한 하드웨어 가속이 요구되고 있다. 현재 벡터 그래픽스 기능을 가장 효과적으로 사용하는 전형적인 사례로는 OpenVG 전용 칩을 이용하여 SVG 를 가속[16]하거나, OpenGL 또는 OpenGL ES 전용 칩을 기반으로 OpenVG, SVG Tiny 를 모두 가속시키는 방법[17]이 제시되어 있다. OpenVG[18]는 휴대용 기기에서 SVG 를 가속하기에 가장 적합한 미들웨어로 평가되고 있으며, 실제로 OpenVG 표준 제정 과정에서, 가장 중요한 목표 응용으로 SVG 모바일 플레이어를 고려 하였다[19].

OpenVG 는 사용자 인터페이스, 벡터 텍스트 렌더링, 전자책, 게임 등의 응용 분야에서의 고품질의 벡터 그래픽스 기능의 가속을 제공하는 것을 목표로 개발 되었다. 최근, 전용의 하드웨어 구현[16]이 시장에 발표되었고, 소프트웨어로 구현된 사례도 있다[20]. 다만, OpenVG 반도체 칩들을 이용하여 일반적인 소비자용 제품을 만들기에는 아직까지 상대적으로 추가적인 비용이 크다는 평가를 받고 있고, 소프트웨어 구현은 비용이 싼 대신, 느린 속도, 즉 충분한 성능을 제공하지 못하는 문제점을 갖고 있다.

본 논문에서는 *OpenVG* 또는 *SVG Tiny* 를 기반으로 개발된 응용 프로그램들을 가속하기 위해서, 다양한 휴대용 기기에서 비트맵 이미지의 처리, 음악이나 동영상 코덱 처리 목적으로 폭넓게 사용되고 있는 멀티미디어 프로세서를 이용하여 가속하는 방법을 제시한다. 본 논문의 실용적 목표 는 고품질/고성능의 벡터 그래픽스 사용자 인터페이스를 제공함과 동시에 CPU 점유율을 낮추는 데 있다. 이렇게 해서 확보된 CPU 처리 능력은 다른 응용 프로그램들의 실시간 수행을 가능하게 함으로써, 대상이 되는 휴대용 기기들의 멀티-태스킹 능력을 향상 시킬 수 있다. 기존의 하드웨어를 사용하여 OpenVG 를 가속하려는 몇몇 시도들이 있었지만, 이들은 주로 OpenVG 전용 칩을 사용하거나, OpenGL ES 칩을 사용해서, 소비자용 제품에서 광범위하게 사용하기에는 높은 비용을 요구 한다.

2 장에서는 기존의 OpenVG 파이프라인을 기반으로 전체적인 하드웨어 가속을 위한 설계와 구현 방법들을 소개한다. 3 장에서는 새로운 설계를 보이고, 4 장에서 구현의 결과들이 제시되고, 이를 바탕으로 5 장에서 결론과 향후 연구 과제를 제시한다.

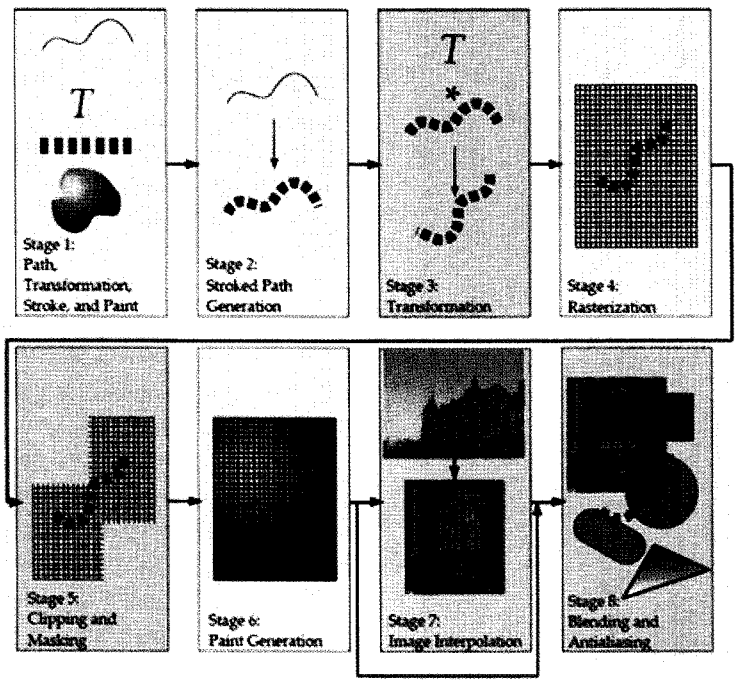


그림 1. OpenVG 의 8 단계 파이프라인 구조

2. 기존 연구

KHRONOS 그룹의 OpenVG 워크그룹은 2004 년 6 월부터 표준화 작업에 착수하여, 2005 년 8 월에 1.0 규격을 공식적으로 발표하였다. 표준과 더불어 Hybrid 사에서 개발한 참조 구현(reference implementation)도 제공되었으나[21], 이의 목적은 규격을 만족시키는 구현이 가능하다는 것을 증명하는 데에 있었고, 실제로 성능이 매우 낮아 실제 상업적으로 활용하는 것은 불가능하였다. 상업적인 소프트웨어를 통한 구현은 ㈜휴원에서 개발하여 2005 년 9 월 발표한 AlexVG engine 이 있다[20]. 다른 연구로는 전자통신연구소에서 임베디드 시스템의 미들웨어 개발의 일환으로 OpenVG 를 구현 하였으며[22], 하드웨어의 구현으로는, 서울대에서 하드웨어 IP 를 개발하여 발표 하였으며[16], 국외에서는 ARM, Imagination Technology, NVIDIA, TAKUMI, Vivante Corp., DMP, Broadcom 등의 회사가 제품을 발표하였다[23, 24].

OpenVG 의 실제 구현에서는 어떠한 구조를 사용해도 문제가 되지는 않지만, 표준 규격에서 규정한 결과를 도출하기 위해서는 OpenVG 표준 규격에서 제시한 8 단계 파이프라인을 참조 하게 된다. 전체를 하드웨어로 구현하거나 소프트웨어로 구현하는 방식과 더불어, 일부를 하드웨어를 사용하고 일부를 소프트웨어를 통해 가속하는 방법으로 OpenVG 를 구현하려는 다양한 시도들이 있어왔다. 대부분은 다른 용도로 개발된 기존의 GPU 하드웨어를 이용하여 OpenVG 를 가속하는 방식을 취했다. 하지만 이러한 구현의 문제점은 GPU 를 추가하는데 상당히 높은 비용이 추가되므로, 고가의 스마트폰에 탑재되나, 저가의 모바일 단말기에서는 널리 쓰이는 칩이 아니며, 이러한 하드웨어의 추가는 기구, 전력 소비 등 추가적으로 고려해야 문제를 만들게 된다. 우리는 이러한 문제를 해결하기 위해서 기존의 많은 이동 단말기에서 동영상 처리 혹은 이미지 촬영 및 처리를 위해 탑재하여 널리 사용되고 있는 멀티미디어 하드웨어를 사용하여 OpenVG 를 가속하고자 한다.

SVG 의 가속에 대한 연구는, 대부분 OpenVG 렌더링 하드웨어로 통합되어, SVG 만의 가속에 대한 문서화된 결과를 찾을 수 없는 상황이다. 본 논문은 OpenVG 와 SVG 를 동시에 의미 있는 가속 성능을 달성하였다는 점에서도 의의를 가진다.

3. 설계 및 구현

OpenVG 의 전형적인 구현은 OpenVG 표준 명세서[18]에 나온 것과 같이, 8 단계로 구성된 파이프라인 구조를 가진다. 그림 1 은 이러한 8 단계에 의한 처리 과정을 보이는 것인데, 각 단계에서의 처리는 다음과 같이 요약할 수 있다.

1 단계에서는 응용 프로그램에서 OpenVG 에서 제공하는 API 를 사용하여 각종 path 의 형태, 색상, 변환, 두께, 패턴 등의 다양한 파라미터들을 정의한다. 2 단계에서는 정의된 path 정보와 스트로크 파라미터들을 결합하여 실제 출력되어야 하는 최종 path 로 재정의한다. 3 단계에서는 변환 정보가 다시 결합되어 새로운 좌표를 계산한다. 4 단계에서 이렇게 계산된 벡터 좌표들을 실제 화면에 매핑한다. 5 단계에서 실제 출력되어야 하는 픽셀들을 결정하고, 6 단계에서 출력될 색상을 결정한다. 이미지를 출력하는 경우에는 7 단계에서 이미지에 의한 출력값이 추가로 결정된다. 8 단계에서 최종적으로 완성된 이미지를 이전의 결과와 합성하여 최종 결과를 얻는다.

OpenVG 의 전체 파이프라인에서 멀티미디어 프로세서로 가속이 가능한 부분은 그림 1 에서 붉은 배경색으로 표시한, 3, 5, 7, 8 단계이다. 전형적인 멀티미디어 하드웨어에서는 다음과 같은 기능들이 제공된다. 각 기능별로 이를 바탕으로 OpenVG 파이프라인을 가속하는 방법을 차례로 설명하겠다.

- **더블 버퍼링(double buffering)과 고속 버퍼 접근(fast buffer access)**: 프레임 버퍼 메모리에서 더블 버퍼링이 가능하다면, OpenVG 하드웨어는 즉시 상당한 가속이 가능하다. 특히, 프레임 버퍼의 부분적인 업데이트를 허용하는 하드웨어 구조라면, 추가적인 가속이 가능하다. 따라서, 더블 버퍼링이 가능한 프레임버퍼에 대한 고속 접근이 가능하다면, OpenVG 의 가속을 위해서는 반드시 사용해야 한다.
- **블렌딩(blending)**: 소프트웨어 OpenVG 구현을 이용하여 다양한 응용 수행 결과를 분석해 보면, 전체 계산량의 50% 이상이 이미지에 대한 블렌딩 연산에서 소비된다. OpenVG 표준 명세에서는 Potter-Duff 블렌딩 [17]와 함께 추가로 몇 가지 블렌딩 모드를 포함하여, 총 10 개의 블렌딩 모드를 정의하고 있다. 반면에, 대부분의 OpenVG 구현에서는 SrcOverDst 블렌딩 모드를 절대적으로 많이 사용하고 있다. SVG Tiny 표준에서는 아예 SrcOverDst 블렌딩 모드만 사용하도록 하고 있다. 이 때문에, SrcOverDst 블렌딩 모드는 저가의 멀티미디어 하드웨어에서도 반드시 제공하고 있으며, 이를 바탕으로, 본 연구에서는 SrcOverDst 블렌딩 모드는 멀티미디어 하드웨어의 SrcOverDst 블렌딩 기능을 반드시 사용하도록 하고, 나머지 블렌딩 모드들에 대해서는 하드웨어에서 제공하지 않는 경우에는 소프트웨어 기능으로 제공하는 방식을 택하였다.
- **이미지 복사(image copy)와 Bit-Blt 기능**: 벡터 그래픽스의 렌더링 과정에서는 의외로 블록 단위의 이미지 복사 기능이 자주 요구된다. 특히, OpenVG 에서 특이하게 제공하는 child image 기능은 근본적으로 블록 단위 이미지 복사가 반드시 필요하다. 따라서, 하드웨어에서 이 기능을 제공한다면, 반드시 사용하도록 하여, OpenVG 와 SVG Tiny 의 전체적인 성능을 상당히 높일 수 있다.
- **리샘플링(resampling)을 동반하는 이미지 변환 (image transform)**: OpenVG 에서는 이중선형(bi-linear) 방식을 포함하여, 다양한 리샘플링(resampling) 모드를 지원하는 어파인 변환(Affine transformation)과 투영 변환(perspective transformation)들을 제공한다. SVG Tiny 의 경우는 어파인 변환만을 제공하고 있다. 어느 경우든, 하드웨어로 이들 변환 과정을 가속시키면 전체 성능이 상당히 좋아진다.
- ***매스킹(masking)과 시저링(scissoring)***: 이 기능들은 화면의 일부에 대해서만 새로운 출력을 하거나, 업데이트하는 경우에 사용된다. 아이콘이나 버튼 이미지를 출력하기에 적합하다. 매스킹 기능은 OpenVG 파이프라인의 4 번째 래스터화(rasterization) 단계나, 다른 앤티-에일리어싱(anti-aliasing) 처리에서도 사용가능하다.
- **컬러 변환(color conversion)**: 이 기능은 SVG Tiny 의 투명성 설정과 이미지 포맷 변환에서 즉각적으로 사용할 수 있다.

대상이 되는 멀티미디어 하드웨어들에서 위의 기능들이 제공되는지 분석한 후에, 우리는 이전에 구현한 완전 소프트웨어 구현[20]에서 해당되는 코드 부분들을 분리했다. 원래의 소프트웨어 구현들을 위의 기능들을 사용하는 부분별로 모듈화한 후에, 완전 소프트웨어 구현을 그대로 사용하거나, 멀티미디어 하드웨어로 가속되는 새로 구현된 모듈을 사용할 수 있도록 선택이 가능하게 하였다.

실제 구현 중에는 위에서 나열한 멀티미디어 하드웨어에 관련된 것들에 추가해서 다음과 같은 기술적 문제점들이 제기되었고, 각각을 해결해서 전체 성능을 높였다.

- **논-스케일링 스트로크(non-scaling stroke)**: SVG Tiny 1.2 에서는 OpenVG 1.1 에서는 제공하지 않는, 벡터의 논-스케일링 스트로크 기능을 제공한다[19]. 이 기능을 지원하기 위해서는 OpenVG 파이프라인 구조에서, 2 번째와 3 번째 단계의 순서를 뒤집어서 수행하는 것이

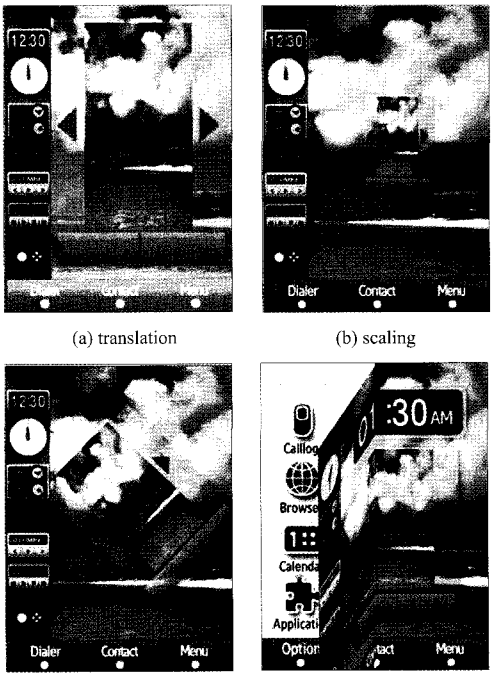
효과적이다. OpenVG 하드웨어를 사용하는 구현들에서는 이렇게 파이프라인 구조를 바꾸는 것이 쉽지 않지만, 본 논문에서 제시하는 모듈화 구조에서는 이것이 가능했고, 결과적으로 기능 제공이 수월했다.

- **렌더링 순서**: 일부 비트맵 처리 가속용 하드웨어들은 비트맵 이미지들의 배치 처리(batch processing)만을 지원하는데, OpenVG 나 SVG 는 각 비트맵이 매번 하나씩 처리된다고 가정하고 있다. 이 때문에 우리는 비트맵 자원의 사용을 최소화하면서도 처리 단계가 서로 싱크로나이즈 되도록 처리하였다.
- **정밀도(precision)**: OpenVG 는 좌표나 각종 파라미터가 실수(floating point)로 처리된다고 가정하고 있다. 반면에, 많은 멀티미디어 하드웨어들은 정수 연산만 가능하거나, 실수 연산이 가능하더라도 유효 자릿수가 상대적으로 작은 경우가 많다. 따라서, 주어진 실수를 내부 처리용으로 변환하면서 오차가 생길 수 있고, 매우 크거나 매우 작은 실수에 대해서는 아예 처리가 불가능할 수도 있다. 우리는 입력된 실수의 범위에 따라, 하드웨어의 사용을 유동적으로 제어해서 이 문제를 해결했다.

4. 구현 결과

본 논문의 구현 결과는 베이스밴드 통신용 칩, 비트맵 프로세서, 이미지 프로세싱 칩, 카메라 제어 칩, DSP 벡터 프로세서 등, 여러 종류의 멀티미디어 프로세서를 대상으로 테스트되었다. 이들을 대상으로, 다양한 OpenVG 응용 프로그램, SVG Tiny 응용 프로그램, 이미지 기반의 사용자 인터페이스 등이 테스트되었다. OpenVG 응용 프로그램들에 대한 테스트 결과는 그림 2 에 제시되어 있다. 그림 3 에서와 같이, SVG Tiny 1.2 인증 검사를 통해 기능 검증에 사용되었다. SVG Tiny 1.2 인증 검사(conformance test)에서는 텍스트 렌더링에 관계된 몇몇 기능을 제외하고는 모두 통과하였다. OpenType 이나 TrueType 과 같은 일반적인 폰트 시스템의 구현은 대부분 윈도우 시스템에 의존하기 때문에 본 연구의 범위에는 포함시키지 아니하였다.

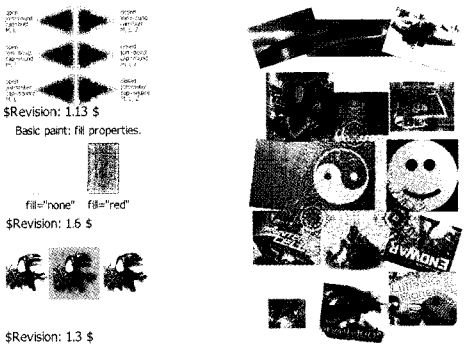
우리는 상업적으로 판매되고 있는 베이스밴드 칩들 중에서도 멀티미디어 가속 하드웨어가 통합되어 있는 모델을 선정해서, 2 차원 그래픽스 가속 기능들을 사용하였다. 즉, 우리는 고가의 스마트 폰 하드웨어가 아니라, 중저가 시스템을 대상으로 하였다. 이들 멀티미디어 처리 시스템은 그림 2 에 제시된 것과 같이, 이미지 블렌딩 기능들과 이미지에 대한 아파인 변환 등, 2 장에서 소개한 각종 가속 기능들을 제공한다.



(a) translation (b) scaling

(c) rotation (d) paging

그림 2. OpenVG 성능 평가에 사용된 프로그램들



(a) SVG CTS (b) 성능 테스트

(a) 그림 3. SVG Tiny 수행 결과

표 1 은 OpenVG 테스트에 대한 렌더링 성능 측정 결과이다. 대부분의 사용자 인터페이스들이 초당 15 프레임 이상의 요구한다는 점에 유의해야 한다. 휴대폰의 화면 크기가 커지면서, 기존의 소프트웨어 구현에서는 CPU 사용률이 100%로 올라가더라도 프레임 재생률이 4~6 프레임 정도로 떨어졌다. 본 논문에서 제시하는 멀티미디어 프로세서를

사용하는 구현에서는 프레임 레이트가 크게 올라갔음을 볼 수 있다. 또한, 화면에 재생해야 하는 콘텐츠의 종류에 따른 프레임 레이트의 변화율(variance)도 줄어들었는데, 이는 다른 응용 소프트웨어들과의 CPU 점유율 경쟁을 피함으로써 나타난 결과이다.

SVG Tiny 플레이어에 대한 테스트는 표 2 에서와 같이, 최대 가속시의 프레임 레이트와, 프레임 레이트를 15 프레임으로 맞춰서 감속시킨 경우를 함께 보였다. 이렇게 감속시키는 경우에는 CPU 점유율이 100%에서 40 ~ 50%로 줄어들면서 실시간 멀티태스킹 능력이 강화된다는 장점이 있다. 테스트에는 그림 3 에서와 같은 SVG Tiny 파일들이 사용되었다.

표 1. OpenVG 기능의 성능 평가

Testing	182MHz CPU with 320 × 240 display				
	(a) full software sol.		(b) multimedia hw		accel. ratio (b/a)
	CPU usage	fps	CPU usage	fps	
Translation	100%	4 – 6	20%	21 – 24	4.9
Scaling	100%	3 – 10	20%	21 – 24	3.5
Rotation	100%	4 – 5	20%	18 – 21	4.3
Paging	100%	2 – 4	20%	22 – 33	8.3
Testing	312MHz CPU with 320 × 240 display				
	(a) full software sol.		(b) multimedia hw		accel. ratio (b/a)
	CPU usage	fps	CPU usage	fps	
Translation	100%	5 – 17	26%	66 - 71	6.2
Scaling	100%	5 – 17	26%	58 – 83	6.4
Rotation	100%	7 – 12	32%	55 – 66	6.2
Paging	100%	4 – 6	23%	125 – 200	29.5
Testing	312MHz CPU with 400 × 240 display				
	(a) full software sol.		(b) multimedia hw		accel. ratio (b/a)
	CPU usage	fps	CPU usage	fps	
Translation	100%	3 – 13	26%	49 – 52	6.2
Scaling	100%	3 – 13	26%	43 – 51	5.8
Rotation	100%	5 – 9	32%	43 – 61	7.2
Paging	100%	3 – 13	23%	92 – 148	24.7

표 2. SVG Tiny에 대한 성능평가

Test Name	Fps	CPU usage	15 fps	CPU usage
animate-elem-06-t	19.85	53.81%	15.00	40.99%
animate-elem-17-t	19.91	51.18%	15.00	39.63%
animate-elem-37-t	19.89	64.69%	15.00	50.16%
render-groups-03-t	15.47	57.32%	15.00	53.82%

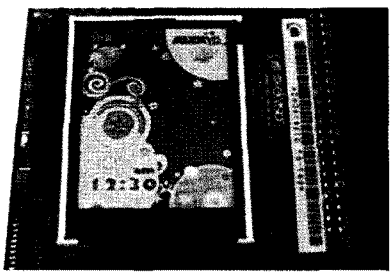


그림 4. 카메라 제어용 칩을 이용한 OpenVG 가속 결과

또다른 예제로는 카메라 제어용 콘트롤러 칩으로 가속시킨 예가 있다. 여기서는 그림 4 에서와 같이, 6M 픽셀 카메라용의 200MHz 짜리 128 배 SIMD 이미지 프로세싱 하드웨어를 사용했고, 20 프레임 이상의 GUI 메뉴들을 그리면서도 10% 이하의 CPU 점유율을 보였다.

5. 결론

우리는 OpenVG 와 SVG Tiny 의 기능들 중에서 멀티미디어 프로세서로 가속하기에 적합한 세부 사항들을 분석하여, 하부의 OpenVG API 와 이를 이용하는 SVG Tiny player 들을 동시에 가속하고자 하였다. 이를 통하여 하부 하드웨어에 적응적으로 가속이 되는 SVG Tiny player 와 이를 효과적으로 지원하기 위해 멀티미디어 프로세서로 가속되는 OpenVG 를 개발하였다. 최종적으로는 기존의 멀티미디어 프로세서들을 이용하여 벡터 그래픽스 기능을 가속함으로써 상대적으로 높은 성능을 달성하였다.

멀티미디어 프로세서를 이용하면 많은 경우에 메모리와 디스플레이 간의 고속 인터페이스가 제공된다는 장점도 있다. 예를 들어, 멀티미디어 코덱 하드웨어를 통한 동영상 재생 기능이나, 카메라 제어용 칩의 뷰 파인더(view finder) 기능을 지원하기 위해, 디스플레이에 대한 고속 업데이트가 가능하다. 이를 이용하면, 저가 휴대폰에서도 멀티미디어 프로세서를 이용한 그래픽스 출력 속도가 더 개선될 수 있었다.

경우에 따라서는 기존의 멀티미디어 기능과, 본 논문이 제안하는 방식의 벡터 그래픽스 가속이 동시에 필요할 수 있다. 예를 들어, 기존의 동영상 재생 기능이 수행되는 화면에서 그 위에 벡터 그래픽스 기반의 GUI 메뉴를 실행하고 싶을 수 있다. 이 경우, 벡터 그래픽스 가속에 멀티미디어 프로세서를 100% 사용할 수는 없다. 이러한 경우를 위한, 멀티미디어 처리 자원의 공유 방법에 대한 연구가 필요하고, 이는 향후 연구 과제로 다루어질 것이다.

감사의 글

본 논문은 지식경제부 지원으로 수행된 "모바일융합 신사업 글로벌 경쟁력강화사업" (과제번호: 10037950)의 지원으로 수행되었습니다.

참고 문헌

[1] W3C consortium, *Scalable Vector Graphics (SVG) 1.1*, second edition, Working Draft, 2010.

[2] W3C consortium, *Mobile SVG Profiles: SVG Tiny and SVG Basic*, 2009.

[3] Open Mobile Alliance, *Wireless Application Protocol Architecture Specification*, version 1.2, 2000.

[4] Open Mobile Alliance, *Multimedia Messaging Service*, version 1.3, 2009.

[5] Open Mobile Alliance, *Dynamic Content Delivery*, version 1.0, 2008.

[6] Java Community Process, *JSR226: Scalable 2D Vector Graphics API for J2ME*, 2006.

[7] Java Community Process, *JSR287: Scalable 2D Vector Graphics API 2.0 for Java ME*, 2009.

[8] Java Community Process, *JSR271: Mobile Information Device Profile 3*, 2009.

[9] ISO, *ISO/IEC 14496-20:2008 – Information technology – Coding of audio-visual objects – Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*, 2009.

[10] W3C consortium, *HTML 5*, Working Draft, 2010.

[11] IDPF, *Open Publication Structure (OPS) 2.0*, Recommended Specification, 2007.

[12] Macromedia Inc., Macromedia flash developer center, http://www.adobe.com/devnet/flash.

[13] Cairo, "Cairo: A Vector Graphics Library", http://cairographics.org/manual.

[14] Silverlight, "The Official Microsoft Silverlight Site", http://silverlight.net/.

[15] Adobe Systems Inc., PostScript Language Reference Manual, 3rd ed., Addison-Wesley, 1999.

[16] D. Kim, K. Cha and S. Chae, "A high performance OpenVG accelerator with dual-scanline filing rendering", IEEE Transaction on Consumer Electronics, 54(3):1301–1311, 2008.

[17] M. Robart, "OpenVG paint subsystem over OpenGL ES shaders", *Digest of Technical Papers International Conference on Consumer Electronics*, 2009

[18] D. Rice, and R. J. Simpson, *OpenVG Specification*, version 1.1, Khronos Group, 2008.

[19] O. Andersson, et.al., *Scalable Vector Graphics (SVG) Tiny 1.2 Specification*, W3C, 2008

[20] H. Lee and N. Baek, "AlexVG: An OpenVG implementation with SVG-Tiny Support", *Computer Standards & Interfaces*, 31(4):661-668, 2009.

[21] Khronos Group, Khronos group home page, http://www.khronos.org/

[22] 이범렬 외, "모바일 3D API 기술 표준화 연구", 전자통신동향분석 제 20 권 제 4 호: 110-119,2005 년

[23] Amanith, AmanithVG GLE, http://www.amanithvg.com/project.html

[24] Hooked Wireless, http://www.hookedwireless.com/OpenVG.html

〈저자소개〉

백낙훈

- 1990년 한국과학기술원 전산학과 학사
- 1992년 한국과학기술원 전산학과 석사
- 1997년 한국과학기술원 전산학과 박사
- 2004년 ~ 현재 경북대학교 컴퓨터학부 교수
- 관심분야: 모바일 그래픽스, 게임 엔진

이환용

- 1990년 한국과학기술원 전산학과 학사
- 1992년 POSTECH 전산학과 석사
- 2011년 경북대학교 전자전기컴퓨터학부 박사
- 2003년 ~ 현재 ㈜휴원 CTO
- 관심분야: 모바일 그래픽스, 영상처리