Khronos OpenVG 1.0 벡터 그래픽 표준 API 구현

이환용*0, 이준영*, 오애경*, 성현찬*, 박기현**
*㈜휴원 솔루션사업부, **계명대학교 정보통신대학
{hylee, jylee, oak, hcsung}@hul.com, khp@kmu.ac.kr

Implementation of Khronos OpenVG 1.0 Standard for Vector Graphics

Hwanyong Lee*0, Junyoung Lee*, Aekyung Oh*, Hyunchan Sung* and Kihyun Park**
*Solution Division, HUONE Inc.
**College of Information and Communication Engineering, Keimyung Univ.

요 약

최근 임베디드 환경에서 2차원 Vector Graphics에 대한 요구는 크게 증가하고 있으며, Flash Lite, SVG등의 응용이 미 널리 사용되고 있다. 반면 이러한 응용을 지원하기 위한 API의 표준은 전무한 실정이었다. OpenVG 1.0은 임베디드 시스템을 위한 미디어 표준 제정 기관인 Khronos Group에서 제정한 2차원 벡터 그래픽스를 위한 API (Application Programming Interface)로 2005년 8월 발표 되었다. 본 논문에서는 OpenVG 표준에 대해 간략히 소개하고 ㈜휴원에서 세계최초로 상용화 개발에 성공한 AlexVG Engine 의 개발과정과 결과에 대하여 설명한다.

[본 연구는 산업자원부 지방기술혁신사업(RT1040-03-02) 지원으로 수행되었음]

1. 서론

최근 모바일 시스템의 성능이 향상되면서 GUI(Graphic User Interface), MMS(Multimedia Message Service), Navigation, SVG Player 같은 다양하고 고수준의 그래픽 환경을 필요로 하는 애플리케이션이 많이 탑재되고 있다. 이런 애플리케이션을 지원하기 위해서는 기존의 임베디드 시스템에서 주로 사용한 텍스트와 비트맵만으로는 처리하는데 어려움이 있다. 따라서, 다양한 그래픽스 기능을 지원하기 위해서 벡터 그래픽스가 대두되고 있다.

벡터 이미지 파일은 그리는데 필요한 좌표, 색상등과 같은 정보만을 가지고 있기 때문에 거의 대부분의 경우 비트맵 이미지에 비해 파일 용량이 작다. 그리고, 비트맵의 경우 애니메이션을 표현하려 할 경우 각 프레임 별로 별도의 이미지를 제작하여야 하는 반면, 벡터 이미지의 경우 하나의 이미지에서 수학적인 좌표 값을 변화시킴으로써 애니메이션을 표현할 수 있으며 비트맵에 비해 자연스럽고 역동적인 애니메이션의 표현이 가능하다. 벡터 그래픽은 확대하거나 축소했을 때 계단현상 등의 이미지 손상이 발생하지 않고 원본 이미지의 품질이 유지하기 때문에 단 한번의 제작으로 디스플레이 장치(LCD)의 사이즈나 해상도에 상관없이 동일한 품질로 사용될 수 있으며, 애니메이션 표현이 매우 용이하기 때문에 각종 모바일 기기용 애플리케이션이나 모바일 서비스에 매우 적합한 기술이다.

현재, 유선 데스크 탑 환경에서는 벡터 그래픽스 표현을 위해 매크로미디어(Macromedia)사의 플래쉬(Flash) 기술과 아도비(Adobe) 회사가 처음 제안한 SVG(Scalable Vector Graphics) 기술이 주로 사용되고 있다. 모바일 환경에서도 이들 두 기술은 각각 Flash-Lite[1]와 SVG - Tiny[2]란 이름으로 모바일 환경에 최적화되어 사용되고 있다. 데스크 탑 환경에서는 해당 기술의 선행 주자인 Flash가 다양한 애니메이션 기능 및 스크립팅 기능을 장점으로 내세우며 SVG 보다 우위를 보이고 있으나, 모바일 환경에서는 비표준 계열로 디자인성을 강조한 Flash - Lite와 더불어 W3C 표준으로 제정되어 있으며 WAP과 연동이 가능한 SVG - Tiny 역시 널리 사용되고 있다.

모바일 시스템에서 벡터 그래픽스에 대한 필요성과 사용이 증가함에 따라 그래픽스 표준 API들을 제정하는 Khronos Group에서 2005년 7월 벡터 그래픽스 랜더링 부분을 표준화하고 가속하기 위한 표준안으로 OpenVG 1.0를 발표하였다. OpenVG는 Flash와 SVG같은 벡터 그래픽 라이브러리들을 위한 하드웨어 가속 인터페이스를 제공하는 공개된 플랫폼 독립적인 API이다.

본 연구의 목적은 OpenVG 표준안을 따르는 소프트웨어 랜더링 엔진 설계 및 구현에 대한 것으로, 이를 위해, 2절에서는 모바일 벡터 그래픽스 관련 기술에 대해 살펴보고, 3절에서는 OpenVG 1.0 표준안을 따르는 벡터 그래픽스 엔진에 대해 살펴본다. 4절에서는 표준안에 따른 벡터 그래픽스 엔진을 구현 검증하며, 5절에서 결론을 맺고, 6절에서는 추후 연구과제에 대해 논한다.

2. 모바일 벡터 그래픽스 관련 기술

2.1 SVG-Tiny

1999년 WWW 컨소시엄은 신축적인 벡터 그래픽스(SVG, Scalable Vector Graphics)라고 부르는 개방된 포맷을 아도비 회사가 주축이 되어 개발하기 시작하였다. WWW 컨소시엄이 2001년 9월에 발표한 SVG 1.0 스팩은 매크로미디어 회사의 플래쉬와 똑같은 그래픽과 애니메이션 기능을 제공할 뿐만 아니라 아래와 같은 장점을 추가로 가지고 있다[2].

첫째, SVG는 WWW 컨소시엄에 의해 표준 자체가 승인되었고, 개방된 포맷이다. 일례로, 모바일 표준 기구인 3GPP는 미국과 유럽에서 사용할 차세대 이동전화를 위한 플랫폼을 정의하는 자리에서 SVG Tiny를 멀티미디어 메시징 서비스(MMS, Multimedia Messaging Service)에 필수적인 포맷으로 채택하였다.

둘째, SVG가 채택한 XML 구문의 포맷은 XML 문서가 가지는 장점을 모두 가지고 있으며, 다른 XML 문서와 통합될 수 있다. 또한 XML DOM 구조에 익숙한 개발자라면 SVG에 접근하기가 훨씬 쉬우며, 웹 서비스와 호환된다.

셋째, SVG는 SMIL 애니메이션 구문을 사용한다. 이 구문은 독립적 SVG 파일에 유용할 뿐만 아니라, SVG 문서가 멀티미디어 SMIL 표현에 내장될 수 있어서 오디오 컴포넌트와 비디오 컴포넌트가 벡터 그래픽 애니메이션과 겹치도록 해준다. 또한 자바 기반의 스크립트를 지원한다. 현재 SVG는 전체 기능을 지원하는 SVG와 모바일용에 최적화된 SVG - Tiny로 구분되어 표준화 작업이 이루어 지고 있으며, 현재 1.2 드래프트(Draft)가 배포되어 있다.

2.2 Flash-Lite

인터넷 환경에서의 벡터 기반 동영상의 사실상의 표준인 플래쉬는 매크로미디어 회사의 제품이다. 이러한 인기를 바탕으로 2003년 2월 매크로미디어사는 Flash 4 스크립팅 엔진 기반의 새로운 Flash 프로필인 Flash - Lite를 발표하였다. 이 프로필은 데스크 탑에서 사용할 수 있는 Flash Player 7의 전체 기능을 지원하기에는 처리 능력과 메모리가 부족한 대중 시장용 휴대폰을 겨냥한 제품이다[1].

Flash - Lite는 일본의 NTT 도코모(DoCoMo)사가 i - mode를 통해 무선 단말기에 서비스하였으며, 국내에서는 디지털아리아가 2002년 최초로 Flash 플레이어를 모바일 단말기에 구현하였다. Flish - Lite 1.1에서는 플래쉬 만의 장점인 액션 스크립팅(Action Script) 4를 지원함과 동시에 아래와 같은 장점을 가진다.

첫째, 네트워크 연결성이 향상되었다. Flash - Lite 컨텐츠는 이제 HTTP 연결을 통해 서버와 데이터를 교환할 수 있으며 휴대폰으로 데이터를 스트리밍 할 수 있어 동적 컨텐츠 업데이트가 가능하다.

둘째, 모바일 SVG 를 지원하여 단일 플레이어로 모든 컨텐츠 재생이 가능토록 하고 있다. 이는 SVG - Tiny가 3GPP에서 의무화 하도록 규정된 이유가 되었을 것이다.

셋째, 네트워크 연결성, 날짜 및 시간, 진동 기능, 언어 지원, 오디오 지원 및 기타 기능을 비롯해 특정 휴대폰 기능에 액세스할 수 있도록 개발자에게 제공한다. 이로 인해 사용자 맞춤형 컨텐츠의 개발이 가능하게 되었다.

넷째, Flash Lite 1.1 CDK(Content Development Kit)를 이용할 수 있다. 개발자들은 이 키트를 사용하여 향후 출시될 플랫폼에 알맞은 컨텐츠를 제작 및 테스트할 수 있다.

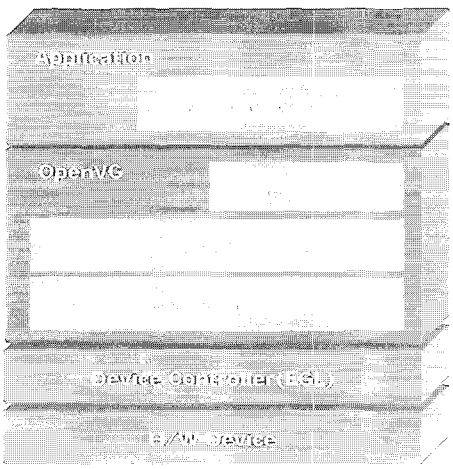
3. 벡터 그래픽스 엔진

본 연구에서는 벡터 그래픽스 처리를 위해 Khronos Group의 OpenVG[3]를 따른다.

3.1 OpenVG overview

OpenVG는 2D 벡터 그래픽스를 하드웨어 가속화를 위한 플랫폼 독립적인 오픈 규격이다. OpenVG는 2D 그래픽스 Application을 개발하기 위해서 사용자에게 제공되는 API들과 API들로 설정된 그래픽스 정보들을 처리하는 이상적인 그래픽스 파이프라인(Pipeline)을 명시하고 있다.

OpenVG의 기본적인 시스템 구성은 [그림 1]과 같은 H/W 상단에 Device Controller(EGL)가 위치하게 되며, 그 다음 벡터 데이터를 랜더링하는 핵심 코어인 Engine, API, UTIL이 위치하게 된다. 그 상위에 Brew, Java 등의 기타 Library를 바탕으로 제작된 Application이 존재한다. OpenVG는 각 디바이스에 종속되는 작업은 Device Controller에서 처리함으로 상위에 구현되는 OpenVG 엔진을 다양한 플랫폼에서 수정 없이 사용 가능하다.

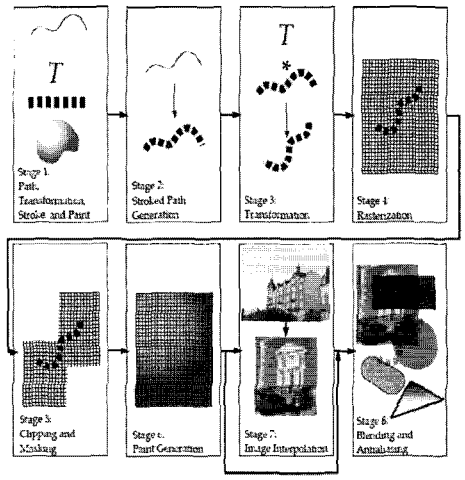


[그림 1] System Architecture of OpenVG[2]

3.2 OpenVG의 Pipeline

OpenVG의 이상적인 그래픽스 엔진은 [그림 2]와 같은 파이프라인(Pipeline)으로 구성되어 있다. Path 정보는 Stage1~6, 8 순서로 처리하고, Image 정보는 Stage1~8 순서로 처리한다.

OpenVG 그래픽스 파이프라인 구현에 있어서 명세서에 따라 처리하는 것이 이상적이지만, 최종 랜더링 결과가 conformance testing process에서 허용하는 정도내에서 명세서에 따라 구현한 결과와 동일하다면 각 Stage들을 이상적으로 따르지는 않아도 된다.



[그림 2] Pipeline of OpenVG[2]

OpenVG 1.0의 각 파이프라인 Stage는 다음과 같은 역할을 하게 된다.

Stage 1 (Define Drawing Parameter & Coordinate) : 사용자가 원하는 좌표, 도형 등을 입력하고, 원하는 드로잉 파라미터를 결정한다. 예를 들면, Stroke하는 선의 두께, Stroke, Fill을 하려고 하는 색상의 결정, 방법의 결정, Transform Matrix의 Loading 등이다.

Stage 2 (Stroked Path Generation) : Path 정보가 stroke되어야 한다면, 입력한 path정보를 일정한 두께의 폭으로 Offsetting을 수행하고 만일 Dash 로 그릴 경우, 이를 반영한 새로운 경로를 생성한다.

Stage 3 (Transformation) : 사용자 좌표 (User Coordinate)로부터 화면 좌표(Screen Coordinate)로 변환하는 과정으로 3x3의 행렬과 벡터간의 곱 연산이다.

Stage 4 (Rasterization) : 현재 Path가 영향을 주는 Pixel

의 coverage value를 구한다. 여기서 구한 coverage value는 저장되어 antialiasing 단계에서 사용하게 된다. 벡터 데이터를 비트맵 데이터로 변환하는 첫 번째 단계이다.

Stage 5 (Clipping & Masking) : 화면의 원하는 영역에만 Display하기 위한 Clipping과 Masking을 수행한다.

Stage 6 (Paint Generation) : Stage 5에서 만들어진 결과물을 이용하여 실제 어떤 색이 칠해질지를 결정하는 단계로 단색의 Fill, 단색 Stroke, Gradation을 이용한 Fill & Stroke, Pattern Fill을 이용한 Fill & Stroke 등을 수행하게 된다.

Stage 7 (Image interpolation) : Image를 처리한다면 이 단계에서 Image에 적용된 matrix의 inverse matrix를 사용해서 interpolating image 값으로 각 pixel의 값을 계산한다.

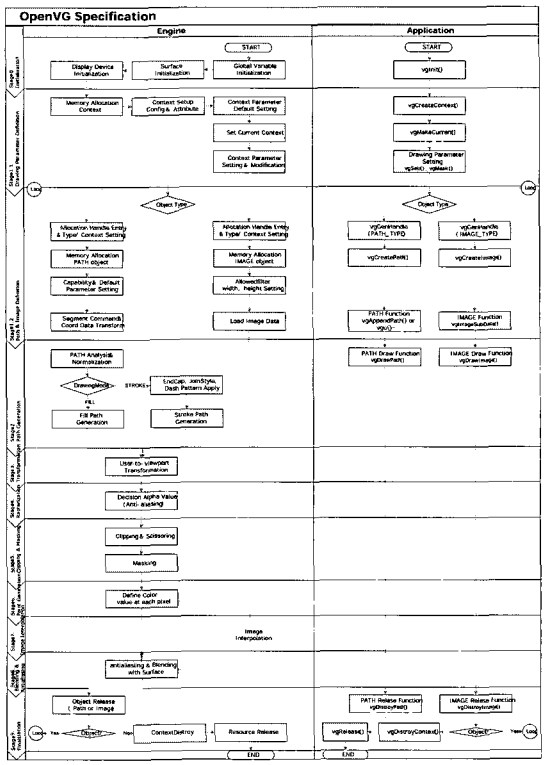
Stage 8 (Blending & Antialiasing) : Stage4에서 계산된 coverage value, 새롭게 생성된 color 값으로 antialiasing된 비트맵 데이터를 생성한다. 이 결과와 이전의 화면을 겹치기 위한 Blending을 한다. Blending 방법에는 Porter-Duff Blending modes, multiply, screen, darken, lighten, additive등이 있다.

이러한 8단계의 Pipeline을 Application 개발 측면에서 분석해 보면 개발자는 OpenVG API를 이용하여 출력하길 원하는 Path 또는 Image등의 Object에 대해 Drawing Parameter를 설정한다(Stage1). 이후 Drawing 함수(vgDrawPath, vgDrawImage)를 호출하면 그 이후는 그래픽스 Engine에서 사용자가 정의한 Drawing Parameter를 분석하여 Object를 구체적인 Bitmap 정보로 변환하고(Stage2~7), 이미 출력된 Bitmap과의 합성이 이루어진다(Stage8). 이러한 과정은 각각의 Object에 대해서 반복 수행해서 한 장의 이미지를 생성하게 된다.

4. 벡터 그래픽스 엔진 구현 및 검증

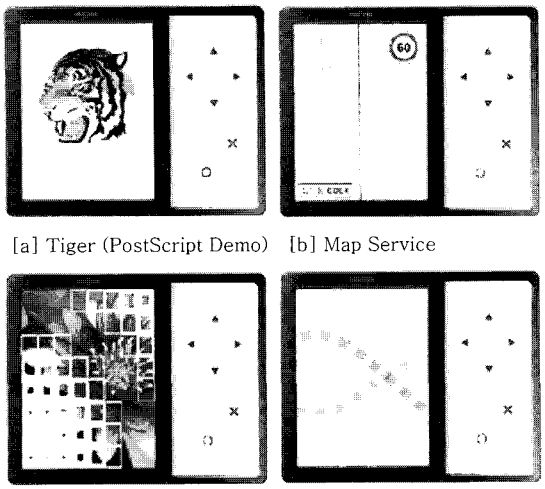
OpenVG Pipeline에 따라 C언어로 벡터 그래픽스 엔진을 구현하였다. 벡터 그래픽스 엔진은 Path, Image Object처리시에 Stage 1~3은 Object 단위로 처리를 하고, Stage 4~8은 처리해야 할 Object의 각 Pixel단위로 처리하도록 구현하였다. 엔진 내부 처리는 성능 향상을 위해서 Fixed Point[4][5][6]로 Q16으로 구현하였다.

[그림 3]은 벡터 그래픽스 엔진의 흐름도를 표시한 것으로 각 Pipeline 단계별 Engine 및 Application에서 수행되는 주요 동작의 상호 관계를 표시하였다. 또한 해당 작업에 실제 호출되는 OpenVG API는 함수명() 형태로 나타내었다.



[그림. 3] OpenVG Pipeline 별 FlowChart

구현한 벡터 그래픽스 엔진을 ARM9 개발보드에 적용하여 Pipeline과 OpenVG 기능 검증하였다. [그림 4]는 개발보드에서 동작하는 응용 프로그램 예시이다



[a] Tiger (PostScript Demo) [b] Map Service

[c]Image Effect(child image) [d] Dash Pattern & Phase

[그림 4] OpenVG Pipeline 별 FlowChart

5. 결론

모바일 단말기의 변화 추세를 볼 때, 프로세스 속도는 점점 빨라지고 출력 장치가 VGA 이상을 지원하며 다양한 크기의 LCD 화면이 보급될 것이라는 점을 감안하면, 비록 래스터(Raster)를 위한 계산량은 비트맵 방식 보다 많지만, 장치 독립적인 벡터 그래픽스 방식으로의 변환은 필수적이라고 생각된다. 이러한 벡터 그래픽스 기술을 적용하기 위해 모바일 단말기에 최적화된 벡터 그래픽스 엔진에 대한 고려가 요구된다.

본 논문에서는 모바일 단말기를 위한 Khronos Group의 OpenVG를 근간으로 벡터 그래픽스 엔진을 구현하였다. 또한, 본 연구에서는 엔진의 기능과 성능을 점검하고 적합성을 확인하기 위하여 검증용 테스트 시스템을 제작하였으며, 이를 통해 구현된 시스템의 성능을 측정하였다. 본 벡터 그래픽스 커널 시스템은 하드웨어와 응용프로그램의 중간 계층에 위치하면서, 초기 목표로 하였던 응용프로그램 및 디바이스에 독립된 커널 시스템으로 동작이 가능하다.

6. 추후 연구 과제

본 기술이 주로 사용될 모바일 단말 환경의 특성상 성능뿐만 아니라 전력소모를 최소화 하는 노력이 필요하다. 이를 위해서는 성능과 비주얼 품질을 알맞게 제어할 수 있는 설계가 필요하다. 또한 아주 작은 수치 오차에 의해서 발생하는 시스템 불안정성을 해결하기 위한 Robustness를 고려한 설계가 필요하다. 또한 본 엔진을 DSP Chip, 3D OpenGL ES 가속 칩 등 다양한 범용 하드웨어 가속 환경에서 활용할 수 있도록 하는 연구가 필요하다.

참고 문헌

[1] Troy Evans," Introducing Macromedia Flash Lite 1.1," Macromedia..

[2] W3C, " Scalable Vector Graphics (SVG) Tiny 1.2 Specification Draft 13," W3C SVG Workgroup, April 2005.

[3] Khronos Group, " OpenVG Specification 1.0" Khronos Group, August 2005.

[4] David Hough, " Applications of the Proposed IEEE-754 Standard for Floating Point Arithmetic" , IEEE Computer, Vol.14, no.3

[5] ARM, " Fixed Point Arithmetic on the ARM" , Application Note 33, ARM, September 1996.

[6] 백낙훈 외, " 고정소수점 수치자료를 사용하는 벡터 그래픽스 연산에서의 오차 전달 모델" , 한국통신 소프트웨어 학회 학술대회, July 2005.