
Hanbit eBook
Realtime 62
2D 벡터 그래픽스 API 표준
OpenVG
프로그래밍
기본편
이환용 지음
한빛미디어
Hanbit Media, Inc.

2D 벡터 그래픽스 API 표준

OpenVG 프로그래밍

기본편

이 책은 지식경제부와 한국산업기술진흥원의 '초광역연계3D융합산업육성사업'의 지원을 받아 출판되었습니다.

2D 벡터 그래픽스 API 표준 **OpenVG 프로그래밍** 기본편

초판발행 2014년 05월 23일

지은이 이환용 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-694-4 15000 / **정가** 13,000원

책임편집 배용석 / **기획** 이중민 / **편집** 정지연
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 이환용, 정지연
마케팅 박상용, 서은옥, 김옥현 / **영업** 김형진, 김진불, 조유미

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 소개

저자_ 이환용

KAIST에서 전산학과를 졸업하고 POSTECH 전산학과에서 석·박사 과정을 이수하였으며 경북대학교에서 박사 학위를 받았다. 20년 넘게 컴퓨터 그래픽스 관련 연구와 개발을 수행해 왔으며, 특히 그래픽스 관련 표준화 활동에 많은 노력을 기울여 왔다.

현재 경북대학교 3D융합기술센터 수석연구원 겸 산학협력중점교수로 재직 중이다. 표준화 기구인 ISO/IEC JTC1 SC24, Khronos Group와 한국정보과학회 CG&I 소사이어티, 한국컴퓨터그래픽스학회, 한국HCI 학회에서 활동하고 있다.

본 도서나 OpenVG 관련 문의 사항은 openvg.programming@gmail.com으로 연락하면 도움을 받을 수 있다.

소스 코드

이 책에 포함된 소스 코드는 다음 주소를 통해 다운받을 수 있다.

● http://www.hanbit.co.kr/exam/2694

상표권 관련 알림

저자 서문

OpenVG는 Khronos Group에서 제정한 2차원 벡터 그래픽스 API 표준으로 모바일 단말기의 GUI 개발, 각종 응용 프로그램 및 서비스 개발에 활용되고 있습니다. OpenVG의 쉽고 명확한 기능과 편리한 사용성 때문에 그 사용 범위가 넓어지고 중요성은 커지고 있습니다.

저전력을 요구하는 단말기의 GUI에 많이 사용되며 그래픽 미들웨어, 웹의 2차원 그래픽스 표준 구현, 벡터 폰트 렌더링, 게임, 전자책 등의 응용 구현을 위한 탁월한 API라 하겠습니다.

이 책은 2010년에 출간하였던 『OpenVG 프로그래밍』(생능출판사,2010)의 오류를 수정하고 좀 더 이해하기 쉽도록 내용을 보완하여 출간하였습니다. 이 책을 통해 OpenVG를 익혀서 유용한 소프트웨어를 편리하게 개발할 수 있게 되길 기대합니다. 이 책에서 잘못된 점이 발견되면 바로 업데이트하여 독자들이 항상 최신의 내용으로 학습할 수 있도록 하겠습니다.

집필을 마치며

저자 이환용

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위해 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

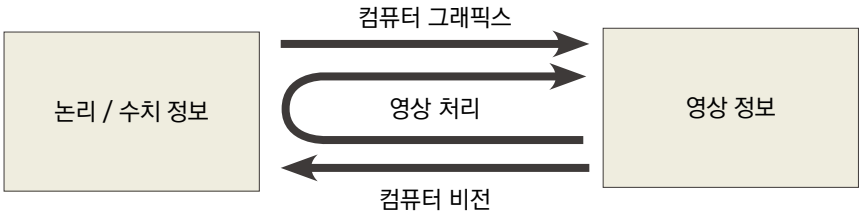
차례

1 | 컴퓨터 그래픽스와 OpenVG

1.1 컴퓨터 그래픽스 개요

컴퓨터 그래픽스Computer Graphics는 컴퓨터를 이용하여 영상을 제작하거나 조작하는 기술을 말한다. 원래 그래픽스는 '도학(圖學)'을 뜻하는 말이다. 그림을 그리는 데 필요한 방법과 기술이 오랜 기간 정립된 학문이라는 뜻이다. 하지만 최근에는 '컴퓨터 그래픽스'를 간단히 '그래픽스'로 표현하는 경우가 많다. 정확히 표현하자면, 컴퓨터 그래픽스는 논리 및 수치 정보로부터 영상을 생성하는 학문으로, 기존의 영상을 처리하여 다른 정보를 갖는 영상으로 변환하는 영상 처리Image Processing와 영상으로부터 유용한 정보를 얻어내는 컴퓨터 비전Computer Vision과는 구별된다.

[그림 1-1] 컴퓨터 그래픽스, 영상 처리, 컴퓨터 비전 비교



1.1.1 컴퓨터 그래픽스의 발전

컴퓨터 그래픽스는 컴퓨터 시스템의 발전과 그 역사를 같이해 왔다. 1950 ~ 60년대에 CRT 모니터와 관련 입출력 장치가 개발되면서 기술 발전의 싹이 트고, 1970년대 들어 각종 컴퓨터 설계 시스템CAD에 컴퓨터 그래픽스가 많이 사용되면서 더욱 발전하였다. 1980년대에는 PC의 보급과 더불어 게임, 애니메이션 등 과학기술 분야뿐 아니라 오락 분야까지 폭넓게 이용되면서 관련 기술이 급격히 발전하여

본격적인 컴퓨터 그래픽스의 황금기를 이루었다. 1990년대 이후에는 기존의 고가 워크스테이션Workstation에서 가능하던 각종 기술이 PC 그래픽 카드로도 가능하게 되어 일반인도 수준 높은 컴퓨터 그래픽스를 이용하게 되었다. 2000년대에 들어서면서 더욱 발전된 컴퓨터 그래픽스는 웹을 통해 널리 보급되었고, 동시에 소형화 기술을 통해 스마트폰, 태블릿 등 개인용 이동 단말기에서도 이용할 수 있게 되었다.

1.1.2 컴퓨터 그래픽스의 분류

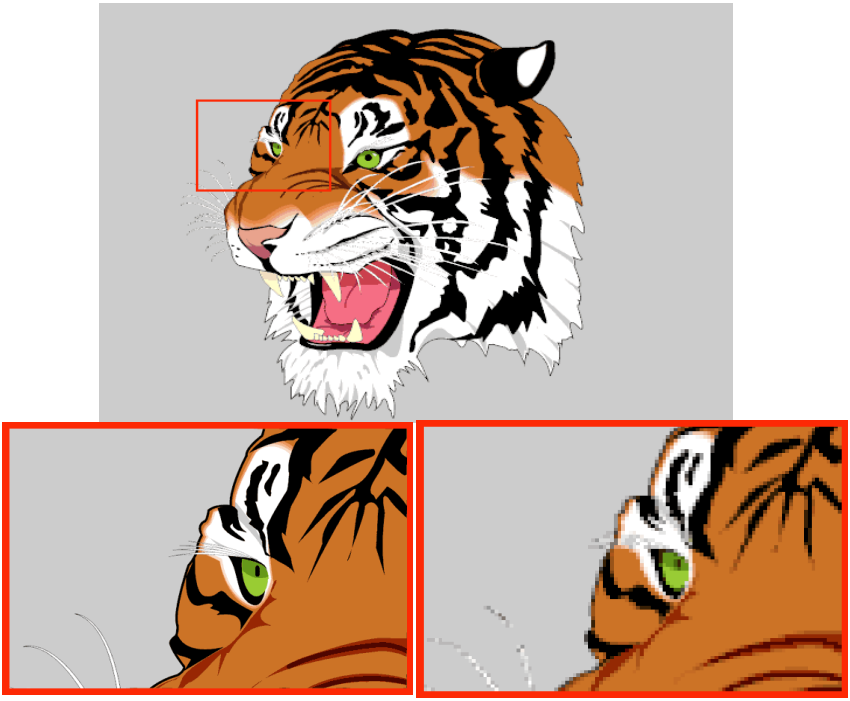
■ 2차원(2D) 그래픽스와 3차원(3D) 그래픽스

2차원 그래픽스는 평면을 다루고 3차원 그래픽스는 입체를 다룬다는 점이 다르지만, 기술적으로는 큰 차이가 없다. 기본적으로 2차원 그래픽스는 입력된 좌표 정보가 X, Y의 2차원 정보로 주어지고 평면 공간에서 변환으로 조작된다. 그에 비해 3차원 그래픽스는 X, Y, Z의 3차원 좌표로 주어지고 3차원 입체 공간에서 변환으로 조작되는 것뿐이다.

■ 벡터 그래픽스Vector Graphics와 래스터 그래픽스Raster Graphics

그래픽스 시스템은 크게 벡터 그래픽스와 래스터 그래픽스로 나누어진다. 벡터 그래픽스에서는 벡터로 정의된 객체Object를 통해 영상을 정의한다. 반면, 래스터 그래픽스(혹은 비트맵 그래픽스, Bitmap Graphics)는 균등하게 나누어진 셀에 색상 정보를 갖는 형태로 저장된다. 보통 접하는 3차원 그래픽스의 대부분은 벡터 그래픽스이며 일부 특수한 의료 영상 장비는 3차원 래스터 그래픽스로 볼 수 있다. 벡터 그래픽스와 래스터 그래픽스는 필요에 따라서 선택적으로 혼용된다. 예를 들면, 래스터 그래픽스 도구인 Adobe사의 Photoshop은 문자와 일부 모양을 벡터 그래픽스로 표현하고 있고, 벡터 그래픽스 도구인 Illustrator는 래스터 이미지를 사용할 수 있는 기능을 포함하고 있다.

[그림 1-2] 벡터 그래픽스(왼쪽)와 래스터 그래픽스(오른쪽) 영상 확대 비교



[그림 1-2]는 2차원 그래픽스에서 벡터 그래픽스와 래스터 그래픽스의 차이점을 보여준다. 그림에서 보는 바와 같이 벡터 그래픽스는 확대하거나 축소하여도 그림의 품질이 떨어지지 않는 장점이 있다.

벡터 그래픽스가 최종적으로 화면에 표시되기 위해서 벡터 정보를 결국 래스터 정보로 바꾸는 과정이 필요한데 이를 래스터화Rasterization라고 한다. 때에 따라서 비트맵으로부터 벡터를 만들어내는 경우가 있는데, 이는 벡터 추적Vector Trace이라고 한다. 이 기능은 대부분의 벡터 그래픽스 소프트웨어에 도구로 포함되어 있다.

[표 1-1] 벡터 그래픽스와 래스터 그래픽스 비교

구분	벡터 그래픽스	래스터 그래픽스
2차원	응용 분야: 디자인, 인쇄 표현 방식: 2차원 벡터 데이터 조합 및 관련 수식 대표 도구: Adobe Illustrator, Corel Draw	응용 분야: 사진·영상 처리 표현 방식: 픽셀(Pixel) 단위 정보의 2차원 Array 대표 도구: Adobe Photoshop
3차원	응용 분야: CAD, 영화, 광고, 가상현실 등 표현 방식: 3차원 벡터 데이터 조합 및 관련 수식 대표 도구: MAYA, SoftImage, 3D MAX 등	응용 분야: 영상처리, 의료, 과학 표현 방식: 복셀(Voxel) 단위 정보의 3차원 Array 대표 도구: 3차원 초음파, 3차원 CT

1.1.3 컴퓨터 애니메이션

애니메이션은 일련의 그림을 빠른 속도로 관객에게 보여주어 움직이는 영상으로 느끼도록 하는 기법이다. 과거에는 조금씩 변화하는 그림을 하나하나 제작하는 셀 애니메이션 방식을 사용하였으나, 근래에는 이를 대부분 컴퓨터 그래픽스를 사용하여 제작한다.

컴퓨터 애니메이션은 컴퓨터 그래픽스의 한 분야로 2차원 그래픽스를 사용하면 2차원 컴퓨터 애니메이션, 3차원 그래픽스를 사용하면 3차원 그래픽 애니메이션이라 칭한다. 이전에는 2차원 컴퓨터 애니메이션은 대부분 기존 만화영화와 같은 그림을 표현하고 3차원 그래픽스 애니메이션은 사실적 렌더링Photorealistic Rendering 기술을 주로 사용한 영상이 대부분이었다. 그러나 최근에는 비사실적 렌더링Non-Photorealistic Rendering 기술의 발달로 셀 애니메이션과 같은 느낌의 영상을 3차원 그래픽스를 이용해서 제작하는 경우가 늘고 있어서 이러한 구분은 의미가 없다.

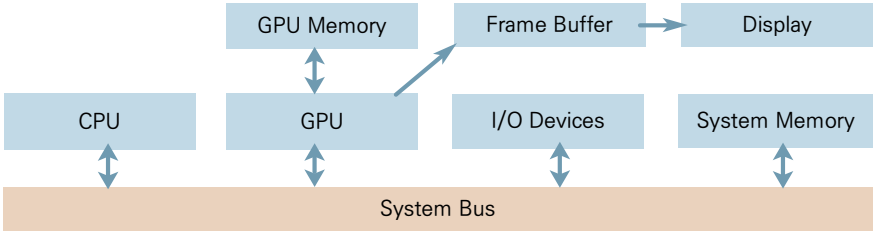
1.1.4 그래픽스 시스템

■ 그래픽스 하드웨어 시스템의 구성

다른 컴퓨터 시스템과 마찬가지로 그래픽스 하드웨어 시스템은 입력 장치, 처리 장치, 기억 장치, 출력 장치의 구성 요소가 있다.

- 입력 장치: 마우스, 버튼, 다이얼, 태블릿 등의 입력 장치
- 처리 장치: 중앙 처리 장치CPU와 그래픽 처리 장치GPU
- 기억 장치: 시스템 메모리, GPU 내 메모리(프레임 버퍼, 기하 데이터 캐시)
- 출력 장치: 디스플레이, 인쇄 장치 등

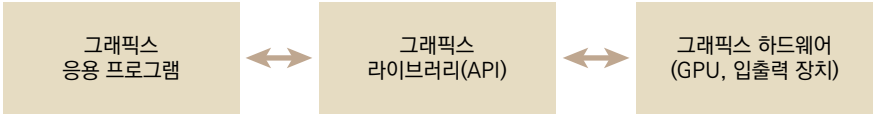
[그림 1-3] 일반적인 그래픽스 하드웨어 시스템의 구성도



■ 그래픽스 소프트웨어 시스템의 구성

그래픽스 소프트웨어 시스템은 그래픽스 응용 프로그램Application Program과 그래픽스 라이브러리로 나누어진다. 그래픽스API, Application Programming Interface는 응용 프로그램을 개발하기 위한 프로그램 라이브러리로 그래픽스 하드웨어와 인터페이스를 담당하며, 그래픽스 응용 프로그램 개발을 위한 다양한 함수를 제공하는 소프트웨어 시스템이다. 여기서 다루는 OpenVG 역시 그래픽스 API 중 하나다.

[그림 1-4] 그래픽 소프트웨어 시스템



■ 그래픽스 API의 기능

그래픽스 API의 기능은 일반적으로 다음과 같다.

- 직선, 다각형, 원, 곡선 등의 그래픽 객체들을 생성하고 출력하는 기능

- 그래픽 객체에 색상, 유형과 같은 속성Attribute 지정
- 그래픽 객체의 모델링, 기하 변환, 시점Viewing 변환, 화면 변환, 색상 변환
- 객체의 렌더링을 통한 이미지 생성
- 그림의 논리 단위(Path, Shape 등) 정의
- 입출력 장치의 제어
- 각종 기능을 편리하게 사용할 수 있는 각종 유틸리티Utility 함수

그래픽스 API는 다양한 응용 프로그램이 다양한 플랫폼에서 사용될 수 있도록 표준화되는 것이 바람직하다. 하지만 그 표준에는 여러 가지가 존재한다. 시장을 주도하는 기업에서 배포하는 API가 사실상 표준 역할을 하거나(Microsoft의 DirectX) 여러 기업이 연합하여 컨소시엄 표준을 만들기도 하고(Khronos Group의 OpenGL) 공적인 표준화 기구(ISO, KS 등)가 국제 혹은 국가 표준을 지정하기도 한다.

단일 기업에 의해 개발된 API는 다양한 기술 지원을 받을 수 있으나, 개발하거나 사용하는 데 비용이 들 수 있고 해당 기업에 대한 의존성이 커지는 단점이 있다. 컨소시엄 표준은 빠른 의사 결정과 시장 진입이 장점이 될 수 있으며 기업 간의 경쟁과 협력이 표준으로 성공하는 열쇠가 된다. 공적 표준은 표준 사용을 강제할 수 있으나, 일반적으로 표준 제정 과정이 복잡하고 시간이 많이 소요된다는 단점이 있다.

1.2 Khronos Group과 OpenVG

1.2.1 Khronos Group

■ 개요

크로노스 그룹Khronos Group은 그래픽스 분야의 몇몇 전문 기업이 다양한 플랫폼에서 리치 미디어Rich Media의 저작과 재생을 위한 표준인 OpenML을 제정하기 위해 2000년에 설립되었다. 이후 더 많은 기업이 참여하고 표준화 작업 분야도 PC와

모바일을 포함한 다양한 플랫폼에서 그래픽스, 미디어의 저작과 재생, 병렬 컴퓨터 프로그래밍, 영상 처리 등으로 확대되었다.

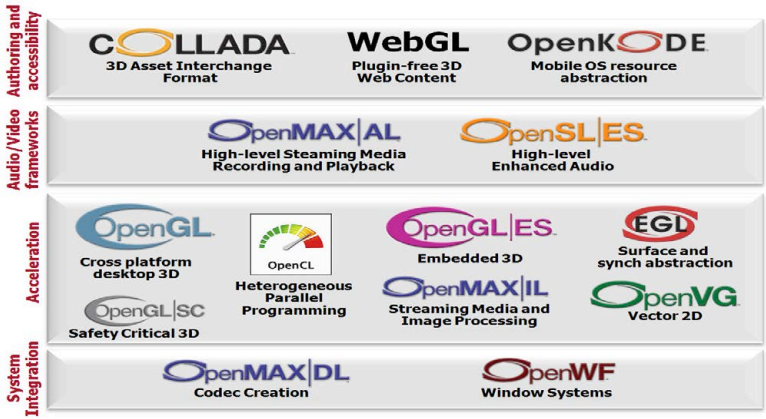
크로노스 그룹은 가입비를 내면 누구든지 가입할 수 있다. 회원 자격은 표준 이사회 역할을 하는 Promoter, 정규회원에 해당하는 Contributor Member, 비영리 기관인 Academic Member, 제정된 표준을 제품으로 개발하는 Adopter Member로 구분된다. 칩 제조사, 단말기 제조사, 솔루션 개발사, 응용 개발사, 연구 기관 및 학교 등 관련 분야의 리더라 할 수 있는 기관이 다수 참여하고 있어서 시의 적절한 표준 제정과 시장 진입을 하고 있다. 우리나라 주요 기관들도 다수 참여하고 있다.

[그림 1-5] 크로노스 그룹 회원사(2014년 1월 현재, 크로노스 그룹 제공)

■ 크로노스 그룹 표준의 종류

크로노스 그룹은 다양한 플랫폼, 특히 모바일 단말기를 위한 미디어 저작 및 재생을 위한 API 표준에 많은 노력을 기울이고 있다. 모바일 단말기 시장이 크게 성장하고 있고 관련 기술이 빠른 속도로 모바일 단말기에 탑재되어 촉망받는 분야이기 때문이다.

[그림 1-6] 크로노스 그룹 표준(2014년 1월 현재, 크로노스 그룹 제공)



[그림 1-6]에서 보는 바와 같이 그래픽스 API 표준으로는 데스크톱 3D 그래픽스 표준인 OpenGL, 임베디드 시스템을 위한 표준인 OpenGL ES, 2D 벡터 그래픽스의 표준인 OpenVG, 사운드에 대한 표준인 OpenSL ES, 미디어 가속 및 서비스를 위한 표준인 OpenMAX가 있고, 3D Asset의 표준인 Collada가 있다. 또한, 병렬 컴퓨터에서 프로그래밍 표준인 OpenCL, 웹에서 3차원 그래픽 사용을 위한 WebGL, 웹에서 OpenCL 사용을 위한 WebCL, 디스플레이 장치와 윈도 시스템 구현을 위한 OpenWF 표준이 있다. 이외에도 최근 영상 처리를 위한 표준 OpenVX, 입력 시스템을 위한 StreamInput, 카메라 인터페이스를 위한 워킹 그룹이 만들어져 표준화를 진행 중이다.

■ 크로노스 그룹의 표준 정책

크로노스 그룹의 표준은 Open/Royalty Free라는 철학 아래 개발된다. 누구든지 거의 무료로 표준을 사용할 수 있고 사용할 때 사용료를 낼 필요가 없다. 가입비를 내고 표준 인증 테스트Conformance Test에 합격하면 제품에 표준 로고를 사용할 수 있다. 이렇게 사용료가 없는 표준을 제정하기 위해서 크로노스 그룹은 독특한 지식

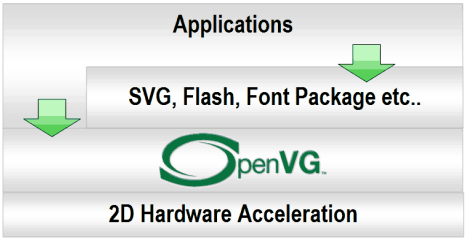
재산권 정책을 수립했다. 제정된 표준이 비회원사의 특허를 침해하지 않도록 노력하며, 회원 기업의 특허일 때에는 회원 기업이 관련 기술을 무료로 제공하도록 독려한다. 이것이 불가능할 때에는 표준에서 관련 기술을 배제함으로써 사용료가 없는 표준이 되도록 하고 있다.

1.2.2 OpenVG 표준

■ 개요

벡터 그래픽스는 컴퓨팅 환경, 특히 웹 콘텐츠의 표시에 폭넓게 사용되고 있다. 예를 들면 Flash, SVG, HTML Canvas는 웹용 벡터 그래픽스로 사용되며, 벡터 폰트는 이미 대부분의 시스템에 널리 사용되고 있다. 반면, 벡터 그래픽스 응용 개발을 위한 API 표준은 존재하지 않았다. 이 때문에 하드웨어 가속 기술도 표준화되지 않아서 많은 벡터 그래픽스 응용들이 하드웨어 가속을 받지 못하는 약점을 갖고 있었다. 이러한 필요에 따라 OpenVG가 2차원 벡터 그래픽스 API 표준으로 정의되고, 하드웨어 가속을 전제로 개발되었다.

[그림 1-7] OpenVG 응용 계층(크로노스 그룹 제공)



OpenVG 표준은 다음과 같은 특징이 있다.

- 낮은 전력 소비: 효율적인 하드웨어 가속은 CPU를 통해 렌더링하는 것보다 전력 소비 효율을 높여줄 수 있다.

- 소프트웨어 렌더링에서 하드웨어 렌더링까지 다양한 구현이 가능: 하드웨어 가속을 위한 표준이나 구현 방식에 어떤 제한도 없다. 따라서 성능 차이를 제외하고는 일관성 있는 렌더링 결과를 얻을 수 있다.
- 확장성Scalability: 벡터 그래픽스의 가장 큰 장점으로, 화면의 크기나 해상도의 변화에 쉽게 대응할 수 있다.
- 현존하는 여러 포맷의 가속: 현재 사용되고 있는 다양한 포맷(SVG, Flash, PDF, HTML canvas, PostScript, Vector Font)의 가속에 활용될 수 있다.
- 다양한 응용 분야: 게임, 지도 서비스, 사용자 인터페이스 개발, 정보 서비스 등 다양한 분야에서 쉽고 빠르게 응용 프로그램을 개발할 수 있다.
- 콘텐츠 이식성: 개발된 콘텐츠는 OpenVG가 탑재된 다양한 플랫폼에 쉽게 이식될 수 있다.
- Royalty Free: 사용료를 낼 필요가 없다.

■ 제공하는 기능

Path

직선, 곡선(2차 및 3차 베지어 곡선), 원호 및 타원 등을 패스Path로 정의할 수 있다.

Transform

OpenVG에서 패스, 페인트Paint, 폰트Font에 대한 변환Transform은 Affine 변환만 지원하고, 이미지에 대한 변환은 3×3 행렬에 의한 Perspective 변환이 가능하다.

Paint

페인트는 패스의 내부를 칠하는 필Fill과 외곽선을 칠하는 스트로크Storke 방법이 있는데, 단색 채우기Solid Color Fill, 선형 및 원형 그라데이션Linear & Radial Gradation, 원하는 이미지 패턴으로 칠하는 패턴 필Pattern Fill 등이 지원된다. 안티에일리어싱Anti-Aliasing에 대한 품질은 사용자가 지정할 수 있다.

Stroke

사용자가 외곽선의 끝 부분End Cap 모양과 연결 부분Join 모양을 결정할 수 있고, 점선Dash도 지원한다.

Image

이미지 파일의 입력 및 이미지 픽셀 형식의 변환이 제공되고 이미지에 대한 다양한 변환도 가능하다. 또한, 이미지 필터를 사용자가 새로 정의하여 사용할 수 있다.

Blending and Masking

이전 영상과 새롭게 렌더링된 결과를 섞어 주는 블렌딩Blending은 다양한 Porter-Duff 함수와 함께 다양한 블렌딩 함수를 지원한다. 또한, 시저링Scissoring과 마스킹Masking 기능을 제공한다.

[그림 1-8] OpenVG 기능 및 활용 예(크로노스 그룹 제공)

■ 구현 및 응용 분야

크로노스 그룹은 표준 문서와 함께 표준 구현에 대해 검증을 할 수 있는 참조 구현Reference Implementation 소스 코드를 공개하였다. 이는 크로노스 그룹 홈페이지[01]에서 내려받을 수 있다.

---

01 http://www.khronos.org/openvg

다양한 방식의 OpenVG 구현 결과가 발표되고 있는데, ㈜휴원은 2005년 9월 최초의 상업적인 소프트웨어 엔진인 AlexVG™ ENGINE을 발표하여 상용 단말기에 탑재하였다. ARM, Imagination Tech, Qualcomm, NVIDIA, TAKUMI, Vivante Group, DMP 등에서도 가속 칩 제품을 출시하였다. 또한, 기존의 3차원 OpenGL ES 가속 칩에서 OpenVG를 가속할 수 있는 기술을 MAZATECH, ㈜휴원에서 발표하였으며, 비트맵 그래픽스 칩의 기능을 이용하여 OpenVG 파이프라인 일부를 가속하는 제품도 ㈜휴원에서 발표한 바 있다.

응용 분야로는 그래픽 사용자 인터페이스GUI 개발에 주로 많이 이용되는데 GUI에서 C 언어로 OpenVG 함수를 호출하는 방식으로 프로그래밍하기도 하고, SVG Tiny, Flash Lite Player를 이용하여 개발하는 예도 있다. 또한, GPS 관련 제품의 맵 드로잉 엔진에 OpenVG를 사용하기도 하고, 일부 폰트 렌더링 솔루션 업체에서는 OpenVG 하드웨어를 위한 폰트 시스템 제품을 발표하기도 하였다.

■ 전망

최근 많은 단말기 생산자들이 OpenVG 탑재에 많은 관심을 두고 있다. 사용자들이 다양하고 미려한 사용자 인터페이스를 요구하고, 사용자 경험User Experience을 강조하고 있는 상황에서 주요 대안이 OpenVG 표준이기 때문이다.

최근에는 타 표준에서도 OpenVG를 많이 사용하고 있다. 예를 들면 MIDP3(JSR271)에서는 일부 벡터 그래픽스 기능을 포함하는데, 이는 OpenVG를 이용하면 쉽게 구현할 수 있다. 그리고 무선 인터넷의 여러 표준(OMA에서 제정한 MMS, DCD, WAP 표준)에서 사용되는 SVG는 OpenVG로 구현하기에 가장 적합한 표준으로, SVG Mobile (SVG Tiny, SVG Basic) 표준은 자바의 JSR226(SVT Tiny 1.1의 Java Binding), JSR287(SVG Tiny 1.2의 Java Binding)에서 사용되고 있다. 또한, Linux의 그래픽 미들웨어인 Cairo는 일부 기능에서도 OpenVG를 지원하고 있으며, Adobe Flash Lite Player도 OpenVG를 지원하고 있다. 최근에는 HTML5의 Canvas Element가 OpenVG를

상당 부분 참조한 표준을 발표하였다.

OpenVG는 앞으로 사용이 증가할 것으로 예상되고, OpenGL에서도 OpenVG의 일부 기능을 포함하는 계획도 발표하는 등 그 중요성이 더욱 커지고 있다.

■ OpenVG의 버전

크로노스 그룹은 2005년 8월 OpenVG 1.0 표준을 공식 발표하였고, 곧이어 일부 내용을 수정한 1.0.1 버전을 발표하였다. 2008년 12월에는 OpenVG 1.1 표준을 발표하면서 폰트 렌더링을 위한 글리프Glyph 렌더링 기능, 멀티 샘플링을 통한 안티에일리어싱, 플래시를 지원하기 위한 기능들을 추가하였다.[02]

1.3 OpenVG 파이프라인

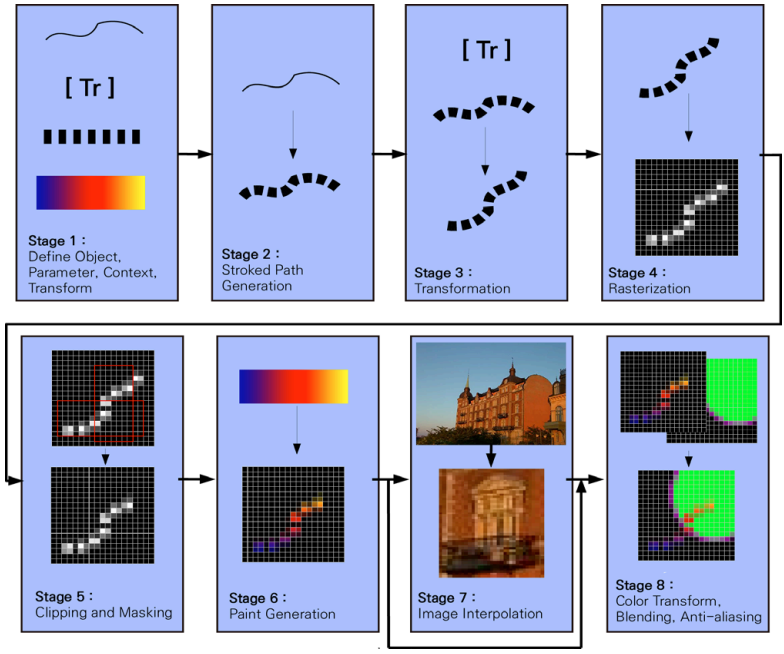
대부분 그래픽 처리 장치는 병렬 처리 기법의 일종인 파이프라인Pipeline 방식으로 구현된다. 파이프라인은 전체 처리 과정에 필요한 단위 처리를 스테이지Stage라 불리는 여러 단계로 만들어 각 스테이지에서 담당한 기능을 처리하고 결과물을 다음 스테이지로 넘기는 방식을 말한다. 실제로 데스크톱에서 사용되는 하드웨어는 파이프라인이 수십에서 수천 스테이지에 이르는 경우도 있다.

OpenVG의 구현 방식을 이해하는 첫 단계는 파이프라인을 이해하는 것에서부터 시작된다. OpenVG 파이프라인을 구현한 결과물을 OpenVG 렌더러 혹은 렌더링 엔진이라고도 한다. 즉, 사용자의 명령에 대해 렌더링 엔진이 이를 처리하여 최종 결과를 드로잉 버퍼Drawing Buffer에 만들어 주고, EGL 명령에 의해 화면에 표시한다.

---

02 본 책에서는 함수의 기능을 설명할 때 특별히 구분이 필요하면 지원되는 버전을 명시한다.

[그림 1-9] OpenVG 파이프라인 예



OpenVG 표준에서는 표준 제정에 기반이 된 파이프라인을 제시하고 있다. 그러나 구현하는 기관이 표준에서 제시하는 파이프라인을 반드시 따를 필요는 없다. 나름의 방법으로 구현한 후 크로노스 그룹에서 제공하는 적합성 테스트를 수행하였을 때 허용된 오차 범위 안의 결과가 나오면 OpenVG 표준을 만족하는 것으로 간주한다. 하지만 OpenVG 표준의 기능과 한계는 파이프라인 구조에 의존하게 된다. 예를 들면 OpenVG에서는 변환할 때 선의 굵기도 함께 변한다. 굵기가 계속 유지되는 선을 그리는 완전한 방법이 OpenVG에는 없는데, 그 이유는 선을 만드는 스트로크 패스 제너레이션Stroke Path Generation 단계가 변환 앞 단계에 오도록 파이프라인이 설계되었기 때문이다. 결국, 선의 굵기를 유지하는 드로잉 기능을 OpenVG 표준에 추가하려면 파이프라인 구조가 일부 변경되어야 한다.

OpenVG 파이프라인은 설정된 콘텍스트Context로 객체를 드로잉 버퍼에 그리는 과정이다. 객체별로 수행한 드로잉 명령마다 한 번씩 이 파이프라인이 수행되어야 한다. 하지만 드로잉 버퍼를 할당받거나 작업이 완료되어 드로잉 버퍼를 화면에 표시하는 명령이 OpenVG에는 존재하지 않는다. 이때 EGL[03] API를 사용하여야 한다.

1.3.1 Stage 1: 정의

스테이지 1은 사용자가 화면에 표시하기 원하는 것의 속성을 정의하는 단계다. 여기에서는 먼저 그리고자 하는 객체의 속성, 이를 그리는 방법을 정의하는 콘텍스트, 칠하는 속성을 정하는 페인트 객체, 그리고 모양이 최종적으로 화면의 어느 위치에 얼마만큼의 크기로 출력될지 결정하는 변환 등 여러 가지 렌더링 품질과 방법에 관해 결정한다.

결국 스테이지 1은 사용자가 드로잉 명령을 내리기 전까지 사용자가 정의한 내용(데이터)을 API를 호출하여 렌더링 엔진에 전달하고 이를 해석하여 렌더링 엔진을 설정Configuration하는 단계다. 드로잉 명령이 호출되면 렌더링 엔진에 의해 파이프라인 스테이지 2 ~ 8이 수행되어 결과물을 만든다.

■ Path, Image, Glyph 객체 정의

OpenVG에서 그릴 수 있는 객체 종류에는 패스, 이미지, 글리프 세 가지가 있다. 패스를 정의한다는 것은 사용자가 그리길 원하는 모양, 즉 객체의 형태를 정의하는 것을 말한다. OpenVG에서 모든 패스는 직선Line과 곡선Curve으로 이루어진다.[04] OpenVG에서는 이미지 렌더링도 이미지 크기에 따라 4개의 꼭짓점을 갖는 직사각형 모양을 패스로 간주하여 처리한다.

---

03 EGL에 대해서는 2장에서 설명한다.

04 이에 대해서는 4장에서 설명한다.

■ Transform 정의

변환을 정의한다는 것은 패스나 이미지 또는 페인트 등의 객체를 화면의 어느 위치에 얼마만큼의 크기로 출력할지를 정의하는 것이다. 이러한 위치, 크기, 회전 등의 변환은 행렬Matrix로 표시할 수 있는데, OpenVG에서는 최종적으로 행렬을 써서 모든 변환을 표현한다.

■ Paint 정의

페인트는 객체를 칠하는 방법을 정의하는 것으로, 2차원 벡터 그래픽스에서는 객체를 칠하는 방법을 적용할 부분이 외곽선과 내부 두 가지로 정의된다. 외곽선을 칠하는 것을 스트로크 페인트Stroke Paint, 내부를 채우는 것을 필 페인트Fill Paint라 하는데 사용자는 자신의 객체에 스트로크를 하거나 필을 할 수 있고 또는 둘 다 할 수도 있다. OpenVG에서 단색 채우기, 선형 그레이디언트Linear Gradient, 원형 그레이디언트Radial Gradient, 이미지 패턴Image Pattern 등의 페인트 방법이 있다.

■ Context 설정

객체를 드로잉하기 위해 다양한 콘텍스트를 설정한다. 여기에는 다음과 같은 내용이 포함된다.

스트로크 관련 외곽선 모양 콘텍스트

- 선의 굵기Line Width: 패스의 스트로크 때 적용할 선의 두께
- 끝 부분 모양End-Cap Style: 패스 끝 부분 모양
- 연결 부분 모양Join Style: 두 개의 세그먼트Segment가 이어져 있을 때 연결 부분 모양
- 점선 설정Dash Array and Phase: 점선으로 스트로크할 때 모양 정의

패스의 품질 및 렌더링 방식 콘텍스트

- 필 규칙Fill Rule: 외곽선으로 구성된 기하의 내부를 필할 때 내부를 정의하는 기준

- 패스 렌더링 품질Path Rendering Quality: 패스의 안티에일리어싱 품질
- 이미지 렌더링 품질Image Rendering Quality: 이미지의 리샘플링 품질
- 픽셀 레이아웃Pixel Layout: 서브 픽셀Sub-pixel 수준의 렌더링을 위한 픽셀의 RGB 패턴 설정

살펴본 것들 외에 렌더링 엔진이 실행되는 과정에 필요한 다양한 파라미터Parameter 값이 있는데 이러한 파라미터들을 콘텍스트 파라미터Context Parameter라 한다. 각 콘텍스트 파라미터의 의미와 세부 사항들은 나중에 자세히 다루도록 하겠다.

1.3.2 Stage 2: Stroked Path Generation

스테이지 2는 정의한 패스의 스트로크를 그릴 때만 수행되는 단계다. 스테이지 1에서 정의한 스트로크 속성에 따라 기하학적 연산을 통해 새로운 패스를 생성한다.

일반적으로 스트로크와 필 두 가지 방법을 통해 렌더링한다. 하지만 OpenVG 파이프라인에서는 두 가지의 다른 방식을 사용하는 것이 아니라, 스트로크 패스 제너레이션Stroked Path Generation을 수행하여 새로운 패스를 만들고, 만들어진 패스를 필하는 방식으로 렌더링하도록 규정되어 있다.

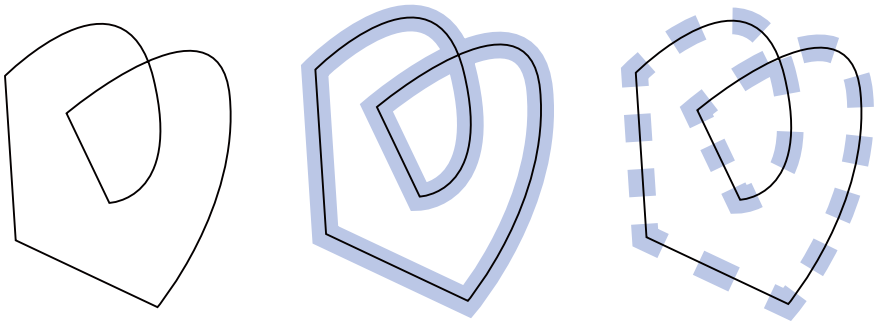
결국 스테이지 2는 입력으로 들어온 패스를 지정된 선의 굵기와 끝 부분 모양, 연결 부분 모양, 점선Dash 정보를 이용하여 새로운 패스를 생성하는 과정이다. .

즉 [그림 1-10]과 같이 사용자가 패스를 정의하거나(a) 두께를 지정하여 단순 스트로크 패스를 생성하거나(b) 대시가 정의된 경우(c)와 같은 결과를 생성하는 것이 이 단계에서 하는 일이다. 이렇게 생성된 패스[05]로는 필을 하는 것과 동일하게 파이프라인의 다음 단계(3 ~ 8)가 수행된다.

---

05 [그림 1-10]에서 회색 부분의 외곽선이 새롭게 생성된 패스다.

[그림 1-10] Stroked Path Generation 예



(a) 입력 Path　　(b) Stroke Path　　(c) Dash Stroke

1.3.3 Stage 3: 변환

본 단계는 스테이지 1에서 지정한 변환이 수행되는 단계다. 스테이지 1에서 사용자에 의해 정의된 필 패스Fill Path나 스테이지 2에서 새롭게 생성된 스트로크 패스Stroked Path의 각 좌표에 대해서 변환 행렬Transformation Matrix을 적용한다. 이 과정을 통해 각 패스는 사용자가 원하는 위치와 크기로 변환된다.

스테이지 3은 입력으로 들어온 모든 패스의 좌표에 대해 변환 행렬을 곱하여 새로운 좌표를 만들고 다음 단계로 넘겨주는 역할을 한다.

■ Affine Transform과 Perspective Transform

그래픽스에서 변환은 크게 Affine과 Perspective 둘로 나뉜다. 기하학적으로 Affine Transform은 평행한 두 선이 변환 후에도 평행이 유지된다. OpenVG에서는 Image ToScreen 변환에 대해서만 Perspective Transform을 할 수 있고 그 외는 Affine Transform만 가능하다.

[그림 1-11] 이미지 변환

(a) 원본 이미지 (b) Affine Transform (c) Perspective Transform

수학적으로 다음 행렬과 같이 Affine Transform은 3×3 변환 행렬의 맨 아래 열이 (0, 0, 1)의 값을 갖고, Perspective Transform은 (w0, w1, w2)의 값을 가진다.

$$\begin{pmatrix} sx & shx & tx \\ shy & sy & ty \\ 0 & 0 & 1 \end{pmatrix} \text{------} \begin{pmatrix} sx & shx & tx \\ shy & sy & ty \\ w0 & w1 & w2 \end{pmatrix}$$

OpenVG 1.0은 PathUserToScreen, ImageUserToScreen, FillPaintToUser, StrokePaintToUser의 4개 변환이 있고 OpenVG 1.1은 GlyphToScreen이 추가되어 5개의 변환이 있다. (w0, w1, w2) 값을 가지는 Perspective 변환은 ImageUserToScreen만 할 수 있다.

1.3.4 Stage 4: Rasterization

스테이지 4는 변환이 적용된 패스가 각 픽셀을 얼마나 차지하고 있는지를 값으로 변환하는 단계이다. 이때의 값을 커버리지 값Coverage Value이라고 한다. 파이프라인의 스테이지 3까지는 벡터 데이터를 처리하고, 스테이지 4에서 래스터화Rasterization가 수행된 이후로는 비트맵 데이터를 처리한다.

픽셀별로 실제 이미지가 차지하는 커버리지 값을 0.0 ~ 1.0(실제로는 0 ~ 255)으로 계산하고, 이후 파이프라인 단계에서 그 수치만큼 값을 반영한다. 이때 안티에일리어싱 처리 여부에 따라 결과가 다르다. 안티에일리어싱을 사용하지 않을 때에는 픽셀의 중심이 패스 혹은 이미지 내부에 있으면 1.0 아니면 0.0의 값을 가지며, 안티에일리어싱을 사용할 때에는 기하 객체가 차지하는 비율을 계산하여 0.0 ~ 1.0의 값으로 저장한다. 이 과정의 정밀도가 안티에일리어싱의 품질을 결정한다.

커버지리 값이 0인 픽셀은 아무런 영향을 주지 못하므로 해당 픽셀은 이후 파이프라인 단계를 수행할 필요가 없다. OpenVG에서는 렌더링 품질 설정하지 않으면 초기값으로 안티에일리어싱을 하는 것으로 설정된다. 따라서 이후 설명할 대부분의 예는 그림(c)와 같이 렌더링된다.

[그림 1-12] Rasterization[06] 예

(a) 입력 Path　　(b) 안티에일리어싱이 없을 때　　(c) 안티에일리어싱을 할 때

1.3.5 Stage 5: Clipping과 Masking

스테이지 5는 크게 클리핑Clipping, 시저링Scissoring, 마스킹Masking의 세 가지 기능으로 구성되며, 그림이 그려질 영역을 지정하는 과정이다.

Clipping

클리핑은 사용자가 지정한 화면에서 벗어나는 부분을 미리 제거하는 과정을 말한다. OpenVG에서는 특별히 클리핑에 대한 규정을 하지 않는다.

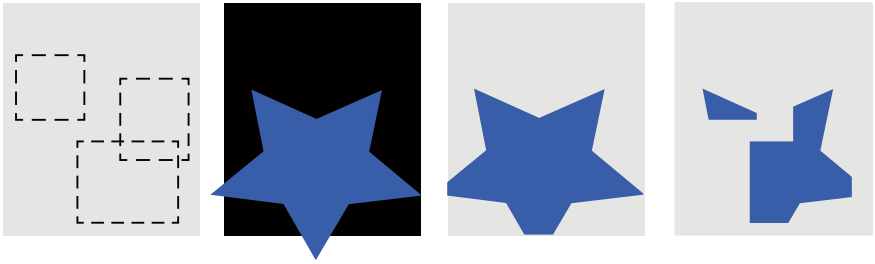
06 검은색은 Path의 내부를 의미한다.

Scissoring

시저링은 사용자가 화면의 특정 직사각형 영역에만 드로잉 결과가 나타나도록 설정할 때 이를 처리하는 것을 말한다. 시저링 직사각형은 스크린 좌표상에서 정의되고, 사용자가 정의한 시저링 영역 외부에 있는 픽셀들의 커버리지 값은 모두 0으로 처리되어 이후 단계에서 무시된다.

[그림 1-13]에서 (a)의 회색 부분이 화면 영역, 점선으로 표시된 사각형들이 사용자가 설정한 시저링 사각형이고, 사용자가 정의한 변환 과정과 래스터화 결과가 별 모양의 (b)라고 할 때 클리핑 과정을 수행하면 화면 밖에 있는 별의 좌측과 하단 일부는 (c)와 같이 잘린다. 여기에 시저링 처리를 하면 사각형 내부 영역에 포함된 값들만 출력된다. 따라서 최종 결과물은 (d)와 같이 그려진다.

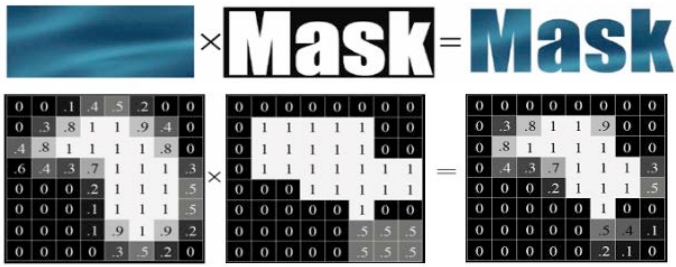
[그림 1-13] Clipping과 Scissoring 예



Masking

시저링이 드로잉을 원하는 영역을 스크린 좌표의 직사각형으로 정의한다면, 마스킹은 픽셀 단위로 결정한다. [그림 1-14]에서 왼쪽 이미지 중앙에 Mask 이미지를 적용한 결과가 오른쪽 이미지다. 이 그림에서는 흰 픽셀이 커버리지 값 1.0을 나타낸다.

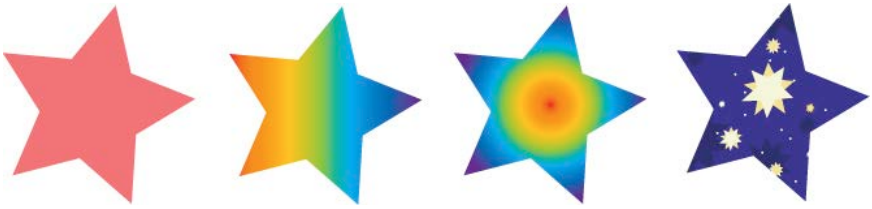
[그림 1-14] Masking 예



1.3.6 Stage 6: Paint Generation

스테이지 6은 스테이지 4에서 결정된 커버리지 값이 0이 아닌 픽셀들에 대해 실제 색상값을 결정하는 단계다. 사용자가 지정할 수 있는 페인트의 종류는 단색 채우기, 선형 및 원형 그레이디언트, 이미지 패턴 등이 있다. 단색 채우기는 각 픽셀이 모두 동일한 색상값을 갖고, 그레이디언트나 이미지 패턴은 복잡한 계산이 요구된다. 특히 패턴은 페인트를 계산하기 위해 영상 처리 기술이 필요하다.

[그림 1-15] 페인트 종류



(a) 단색 (b) 선형 그레이디언트 (c) 원형 그레이디언트 (d) 패턴

1.3.7 Stage 7: Image Interpolation

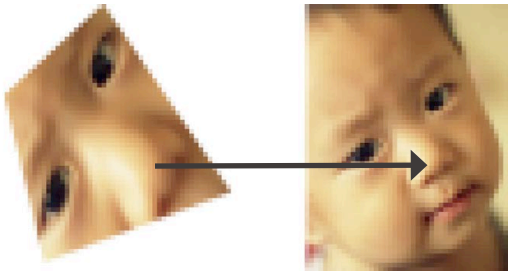
스테이지 7은 이미지가 관련되었을 때, 즉 이미지를 렌더링할 때(또는 패스에 이미지 패턴으로 페인트할 때) 수행되는 단계로 각 픽셀에 해당하는 이미지 색상값을 원본 이미지 데이터로부터 가져오는 단계이다. 일반적으로 해당하는 변환 행렬의 역행렬

을 구해서 행렬과의 곱셈으로 처리한다.

이때 일반적으로 이미지의 리샘플링Re-sampling이 일어나는데 리샘플링 품질은 이미지가 생성될 때 지정되었던 이미지 품질을 따른다.

이렇게 구해진 픽셀별 이미지 색상값은 이미지 출력 형식이 Normal일 때는 해당 이미지 값이 바로 최종 픽셀의 색상값이 된다. 즉, 스테이지 6의 결과물인 페인트 색상값은 무시된다. 만일 이미지 출력 형식이 Multiply 또는 Stencil이면 스테이지 6에서 결정한 각 픽셀의 페인트 색상값과 조합되어 해당 픽셀의 최종 색상값이 된다. 스테이지 7까지 완료하면 객체는 사용자가 설정한 사항에 따라 렌더링이 완료되어 이미지[07]가 만들어진다

[그림1-16] Image Interpolation



1.3.8 Stage 8: Color Transform, Blending and Antialiasing

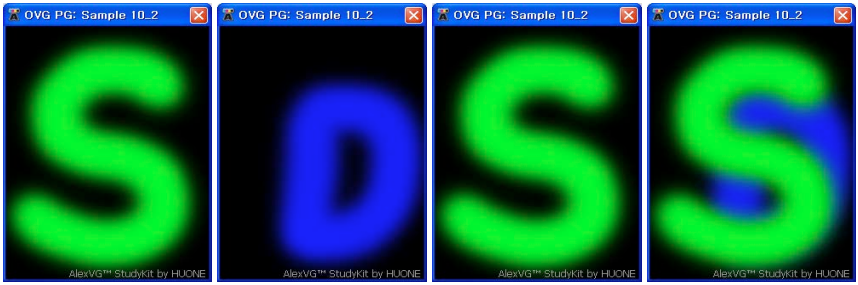
스테이지 8은 스테이지 7까지 수행하여 만들어진 소스 이미지Source Image, src를 실제 드로잉 버퍼인 대상 이미지Destination Image, dst에 병합하는 단계로, 이 과정을 블렌딩Blending이라고 한다. 다양한 블렌딩 방법이 있으며 이에 따라 최종 이미지 결과도 달라진다. 블렌딩한 최종 결과 이미지는 다시 파이프라인이 수행될 때 대상 이미지로 사용된다.

---

07 이것을 소스 이미지(Source Image)라고 한다.

또한, 이 단계에서는 스테이지 4에서 계산된 커버리지 값을 사용하므로 실제로 안티에일리어싱을 수행하는 단계라고 할 수 있다. 그리고 OpenVG 1.1부터 새롭게 추가된 기능인 색상 변환을 통해 블렌딩 직전 소스 이미지의 색상과 투명도를 조정할 수 있다.

[그림1-17] Blending 예



(a) Src(Green) (b) Dst(Blue) (c) Src Blending (d) Src Over Blending

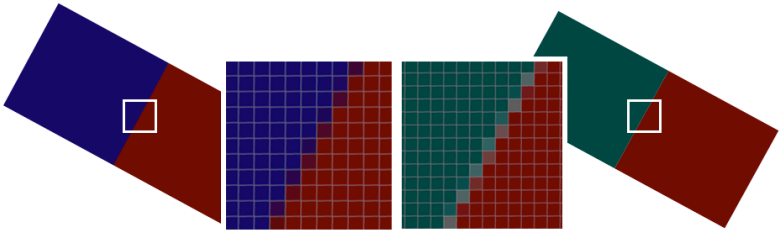
1.3.9 Stage A: Multisampling(OpenVG 1.1 Only)

스테이지 A는 표준에 정의되어 있지만, 선택적으로 사용할 수 있다. 멀티샘플링 Multisampling은 안티에일리어싱 방법의 하나로 해상도를 n배 높여서 작업한 후 이를 1/n로 줄여서 출력하는 방식이다. 1/n로 줄이는 과정은 각각의 객체를 렌더링할 때 실행하는 것이 아니라 객체를 모두 n배로 렌더링한 후 최종적으로 디스플레이 되기 직전에(eglSwapBuffer 명령이 실행될 때) 실행한다. 따라서 객체 단위로 실행되는 기존의 파이프라인 8단계와 구분된다.

따라서 멀티샘플링은 EGL 표준의 지원을 받아야 한다. 이 방식은 [그림 1-18]과 같이 두 개의 인접한 패스의 렌더링에서 패스 렌더링마다 각각 안티에일리어싱을 했을 때(오른쪽 이미지) 가운데 확대 그림처럼 바탕의 흰색이 스며 올라오는 현상 Bleeding이 일어나는 것을 줄일 수 있다는 장점이 있다. 또한, 해상도를 n배 높인 것 외에는 특별한 알고리즘이 필요 없어서 구현이 단순하다. 하지만 해상도를 n배 높

이는 작업을 실행해야 하므로 성능이 낮아지고 전력소비가 커진다.

[그림 1-18] 멀티샘플링 안티에일리어싱의 효과



1.3.10 정리

지금까지 OpenVG 파이프라인의 각 단계와 그 기능에 대해서 살펴보았다. OpenVG는 8개의 파이프라인으로 구성된다. 스테이지 1에서는 사용자가 원하는 사항들을 항목별로 정의하고, 정의된 내용은 즉시 또는 렌더링 직전에 렌더링 엔진으로 전달된다. 그리고 드로우 명령(vgDrawXXX)을 실행하면 스테이지 2 ~ 8의 각 단계가 실행된다.

- 패스의 스트로크가 있을 때만 스테이지 2에서 새로운 패스를 생성한다.
- 사용자가 패스에 대해 필과 스트로크를 동시에 지정하면 필과 스트로크 두 차례 파이프라인을 수행한다. 이때 필이 먼저 수행되고 이후에 스트로크가 수행된다.
- 이미지는 기본적으로 직사각형 모양 패스의 필과 동일하게 파이프라인을 수행하고, 추가로 스테이지 7을 수행해야 한다.
- 이미지는 이미지 출력 형식이 Normal이면 스테이지 6을 수행할 필요가 없다.

이 밖에 OpenVG 표준 관련 제품 및 교육 자료 등 새로운 자료가 필요하면 다음 사이트를 방문하기 바란다.

- 크로노스 그룹 관련 정보 및 뉴스: http://www.Khronos.org/
- OpenVG 관련 정보 및 참조 구현 다운로드: http://www.Khronos.org/openvg