Interactive TV를 위한 Polygon 기반의 3D Graphic Engine

*김정환, *정문열
서강대학교 영상대학원 미디어공학과 디지털방송랩
ninei@sogang.ac.kr, moon@sogang.ac.kr

Polygon based 3D graphic engine for interactive TV

*Jung-Hwan Kim, *Moon-Ryul Jung
Digital Broadcasting Lab of GSMC at Sogang Univ.

요 약

디지털 방송은 비디오, 오디오와 더불어 데이터를 전송할 수 있다. 데이터 영역에는 셋톱박스(STB)에서 수행되는 애플리케이션(Xlet)이 포함된다. 디지털 방송은 애플리케이션을 이용해 보다 진보된 인터랙티브 프로그램을 제공할 수 있다. 그러나 이러한 인터렉티브 TV에서 3차원 객체의 활용은 아직 여러 제한사항으로 인해 어려움이 있다. 현재 제정된 DVB-MHP, SCTE-OCAP, ATSC-ACAP 등의 데이터 방송 표준은 3D 객체를 표현하기 위한 구체적인 방법을 기술하지 않고 있다. 또한, 상용 셋톱박스에 탑재된 자바 가상 머신(Java Virtual Machine)은 3D 객체를 렌더링(Rendering)하기 위한 API를 제공하지 않고 있다. 그리고 이를 위한 별도의 플러그인(Plug-in)도 제공하지 않는다. 본 논문에서는 Interactive TV에서 3차원 객체를 표현하기 위해 필요한 기본개념과 기술 및 데이터 방송 표준을 분석하고, 이들을 통합하여 polygon 기반의 3D Graphic Engine을 제안한다. 특히, 본 논문에서 제안하는 3D Graphic Engine API는 각각의 표준에서 가용한 공통 API를 기반으로 설계 되었으며, Java Virtual Machine 1.1 환경에서 구동될 수 있도록 구현되었다.

Ⅰ. 서 론

디지털 방송은 다채널 고화질 서비스 이외에도 데이터방송 서비스를 제공한다. 데이터방송은 비디오, 오디오와 함께 데이터를 전송하여 방송하는 것을 말한다. 데이터방송에서 전송되는 데이터에는 셋톱박스(STB)에서 구동되는 애플리케이션(Xlet)과 이미지, 사운드, 텍스트 등의 파일이 포함된다.

데이터방송에서 애플리케이션(Xlet)은 방송 프로그램이 시청자에게 좀더 다가갈 수 있는 촉매제로서 인터렉티브 프로그램 제작에 중요한 요소이다. 또한, 방송프로그램을 제공하는 방송사와 시청자간에 상호작용을 유도하며 양방향 방송을 가능하게 한다. 하지만, 인터랙티브 TV에서의 Xlet은 2차원의 평면 이미지를 이용한 애플리케이션이 거의 대부분이다. 2차원 객체보다 사실감, 존재감, 입체감 등의 장점을 지닌 3차원 객체는 여러 제한사항으로 인해 그 활용이 아직 미흡하다.

각각의 데이터 방송 표준들은 3D 객체를 표현하기 위한 명확한 정의나 기술을 구체적으로 기술하고 있지 않다. 또한, 각각의 표준에서 정의되어, STB에 탑재된

JVM은 자바에서 제공하는 3D 관련 API를 사용하기엔 너무 낮은 버전이다. STB에서 별도의 플러그인을 제공하는 것도 아니다. 즉, Xlet을 위한 3D 관련 API가 거의 없는 실정이다.

본 논문은 이러한 제한사항을 극복하고 인터랙티브 방송프로그램 제작과 다양한 디지털방송 서비스에 기여할 수 있는 Polygon 기반의 3D Graphic Engine을 제안한다. 본고에서 제안하는 3D Graphic Engine은 그 활용에 따라 20~40kbyte의 작은 용량을 가지며, STB에 Xlet과 함께 전송되어 실행된다. 또한, 각각의 데이터 방송에서 지원되는 공통API와 JVM1.1 환경에서 구동되도록 설계되어, 현재 제정된 각각의 데이터방송 표준에 호환 될 것이다. 더불어, 범용 라이브러리인 OpenGL과 유사한 API를 제공함으로써, Xlet 프로그래머가 좀더 쉽고 빠르게 익히고 응용할 수 있다.

본 논문의 진행 순서는 다음과 같다. **Ⅱ장 애플리케이션, 데이터 방송 공통API 분석**에서는 애플리케이션(Xlet)의 구동환경과 life cycle, 각각의 데이터방송 표준에서 추출하여 선별한 공통API를 기술한다. **Ⅲ장 Polygon 기반의 3차원 객체의 표현**에서는 3D 객체를 표현하기 위한 기본 적인 이론을 설명한다. 이어서 **Ⅳ장 3D Graphic Engine 구현**에서는 Polygon 기반의 3D 객체의 구성과 3D Graphic Engine API를 설명한다. **Ⅴ장 실험 및 결론**에서 실험 결과를 토대로 본 논문의 결론을 맺고 추후 연구과제를 도출할 것이다.

Ⅱ. 애플리케이션, 데이터 방송 공통API 분석

본 장에서는 본 논문의 이해를 돕기 위해 STB에서 구동되는 애플리케이션(Xlet)의 구조, 각각의 방송규약에서 가용한 공통 API를 설명한다.

1. 애플리케이션(Xlet)의 구조

디지털 방송에서 애플리케이션은 STB에서 수행되는 자바 프로그램으로 Xlet이라 불린다[1][4]. 디지털 방송은 Xlet을 이용해 T-Commerce, T-Game 등의 다양한 서비스를 제공한다.

Xlet은 셋톱박스의 미들웨어(운영체제 위에서 돌아가는 시스템 소프트웨어)가 구현하는 API를 기반으로 구현되며, data/object carousel[1][5]에 포함되어 MPEG-2 TS[7]를 통해 전송된다. 전송된 Xlet이 셋톱박스에서 수행되기 위해서는 Xlet을 관장하는 매니저가 필요한데, 셋톱박스의 애플리케이션 매니저(Application Manager)가 그 역할을 수행한다.

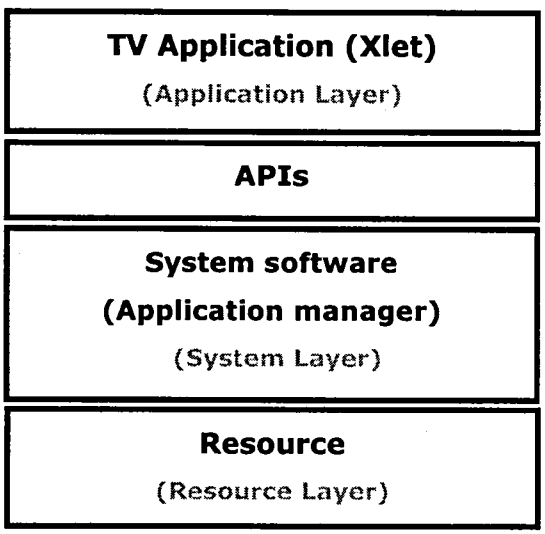

그림1. 셋톱박스의 Xlet 구동 환경

애플리케이션 매니저는 Xlet의 loaded, paused, active, destroyed의 4가지 상태(state)를 제어한다. Xlet 프로그래머는 이를 위해 initXlet(), startXlet(), pauseXlet(), destroyXlet()을 반드시 정의해야 한다. 그림 2는 Xlet life cycle을 도식화한 그림이다.

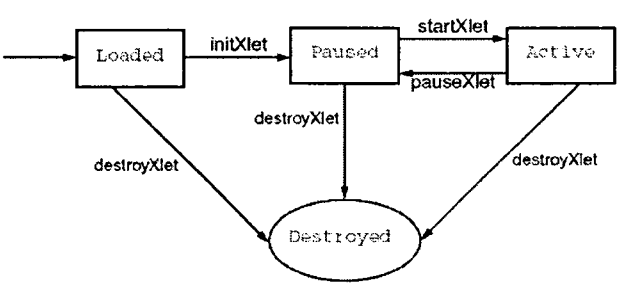

그림2. Xlet life cycle [1]

2. 데이터 방송 공통API 분석

Xlet은 STB에서 구동되기 위해 미들웨어가 제공하는 API를 이용한다. 이러한 API는 유럽의 DVB-MHP[1], 북미의 SCTE-OCAP[2], ATSC-ACAP[3] 등의 데이

터 방송 표준에서 정의하고 있다.

각각의 데이터방송표준 API는 지원여부에 따라 크게 범용API, 조건부API, 특성화된 API로 구분될 수 있다. 첫째, 범용API는 모든 표준에서 공통으로 사용되는 API이다. 둘째, 조건부API는 각각의 표준에서 일부만 지원되는 API이다. 마지막으로 특성화된 API는 특정 표준에서만 지원되는 특성화된 API이다.

예를 들자면, javax.tv는 모든 표준에서 동일하게 사용할 수 있는 범용API이다. org.davix.media는 MHP에서는 모두 지원되지만, 다른 표준에서는 제한적으로 쓰이는 조건부API이다. 마지막으로, org.davic.mpeg.dvb는 MHP에서만 지원되는 특성화된 API이다. 범용API가 아닌 조건부 API나 특성화된 API의 사용은 표준간의 호환성에 걸림돌이 된다.

구 분	범용 APIs
Personal Java 1.2	java.awt, java.io, java.lang, java.util
Java TV[6]	javax.tv

표1. 3D Graphics Engine APIs

이러한 문제점을 극복하고 표준간의 호환성을 위해 각각의 표준에서 모두 지원되는 범용API(표1)를 추출하여 선별하였다. 본 논문에서 제안하는 3D Graphics Engine은 JVM1.1과 표1의 API를 기반으로 설계되었으며, 실험을 통해 MHP, OCAP STB에서 확인하였다.

Ⅲ. Polygon 기반의 3차원 객체표현

본 장에서는 3D 객체를 표현하기 위한 기본적인 요소와 3D Graphic Engine의 가상좌표계를 소개한다. 또한, polygon을 기반으로 생성한 3차원 객체의 구성원리를 간략히 설명한다.

1. 3차원 객체의 표현 요소

3차원 객체를 표현하기 위해 중요한 3가지 요소는 광원, 카메라, 사물이다. 이는 실제세계에서 인간이 사물을 인지하는 요소이다. 광원은 태양이나 조명, 카메라는 사람의 눈, 3차원 객체는 실제 보여지는 사물에 해당된다. 또한, 3차원 객체를 렌더링하기 위해서는 평면의 x, y축과 더불어 깊이(depth)를 나타내는 z축이 필요하다.

2. 3차원 객체를 위한 가상좌표계

디지털방송은 화면구성을 위해 graphics reference model을 정의하고 Background planes, Video planes, Graphics planes을 제안한다[1].

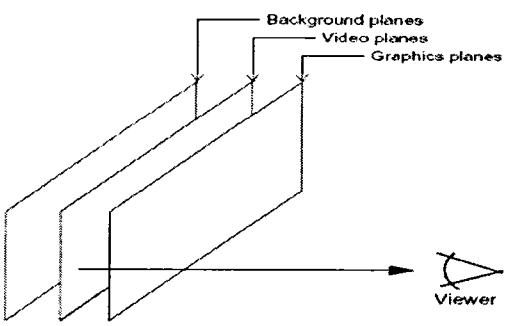

그림 3. Graphics reference model planes [1]

본 논문에서는 Graphics planes(그림3 참조)을 근간으로 2차원의 planes에 임의의 z축을 설정하고 가상의 3차원 좌표계(그림4)를 생성한다.

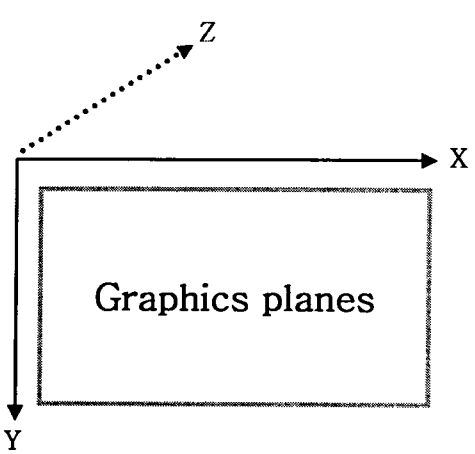

그림 4. 3D Graphic Engine 가상 좌표계

3D Graphics Engine은 가상의 3차원 좌표계(그림4)를 기준으로 카메라와 광원의 위치를 결정하고, 각각

의 축을 기준으로 이동과 회전이 가능한 3차원 객체를 표현한다. 3차원 객체의 이동과 회전은 벡터(vector), 행렬(matrix)을 이용해 구현된다.

3. Polygon 기반의 3D 객체구성

Polygon이란 삼각형이나 사각형, 오각형 등과 같이 세 개 이상의 변을 가지고 있는 닫힌 도형을 일컫는다. 3차원 컴퓨터 그래픽스에서는 이들 다각형을 조작해서 입체 도형을 구성한다.

본 논문에서는 제안하는 3D Graphic Engine은 가장 기본이 되는 삼각형 polygon을 기반으로 모든 객체를 구성한다. 예를 들어, 사각형은 두 개의 삼각형 polygon으로, 원은 다수의 삼각형 polygon의 조합으로 표현할 수 있다. 이러한 원리로 box, sphere, cylinder 등의 3차원 객체를 구성한다. 그림5는 3D Graphic Engine이 삼각형 polygon으로 구성한 3D 객체이다.

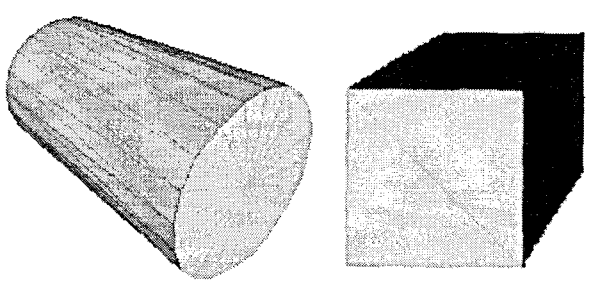

그림 5. 삼각형 polygon 기반의 오브젝트 구성

그림5에서 보면, 다수의 삼각형 polygon으로 구성된 3D box는 화면에 보이지 않는 영역이 존재함을 알 수 있다. 3D Graphic Engine은 이러한 영역을 감지하고, 다수의 3차원 객체가 겹쳐져있는 경우 Z축을 기준으로 정렬한 후 보여지는 부분만 렌더링 하도록 설계되었다.

Ⅳ. 3D Graphic Engine 구현

본 장에서는 3D Graphic Engine의 구조를 설명하고, 예제코드를 통해 API를 간략하게 소개한다.

1. 3D Graphic Engine 구조

본 논문에서 제안하는 3D Graphics Engine의 구조(그림6)를 클래스 위주로 살펴보도록 하자.

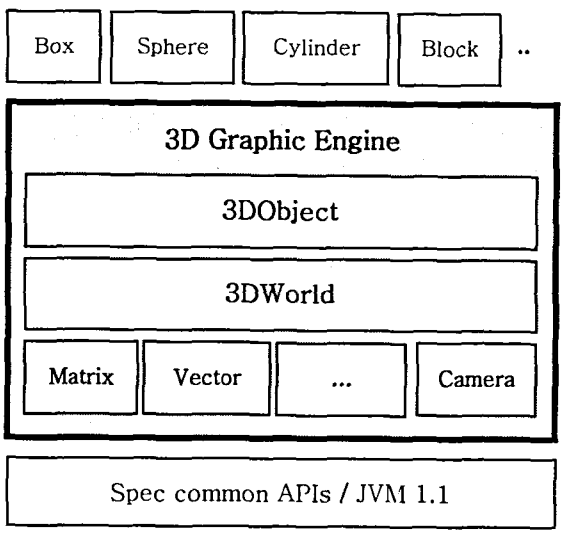

그림 6. 3D Graphic Engine 구조

Engine에서 가장 기본이 되는 클래스는 Matrix, Vector, Polygon, Camera 등이다. 이들은 3차원 좌표계, 광원, 카메라, 오브젝트 등을 구성하는데 근간 클래스가 된다. 이러한 뿌리 클래스를 바탕으로 그 상단에 3DWorld 클래스가 놓여진다. 3DWorld 클래스는 3차원 좌표계와 광원, 카메라를 설정한다. 또한, 각각의 객체를 관장하며, 3차원객체를 2차원 평면으로 투영한다. 3DWorld에 의해 관장되는 각각의 모든 3D 객체들은 부모 클래스인 3DObject 상속받아 구현될 수 있다. Box, Sphere, Cylinder, Block등은 실험에 사용되었던 기본객체이다.

특히, 그 기본객체들은 새로운 형태의 3D 객체를 생성할 때 사용될 수 있다. Block은 Box를 상속받아 구현되었으며, 실험을 위한 3D-Tetris 게임의 기본 구성 요소로 사용되었다.

2. 3D Graphic Engine의 Sphere 구현

3D Graphic Engine의 초기화를 위해 x, y, z의 3가지 축을 지니는 가상의 3차원 좌표계(그림4 참조)를 설정하고, 광원과, 카메라의 위치를 결정한다.

본 논문에서 제안하는 3D Graphic Engine의 API를 이용한 약식코드와 3D Sphere(그림7) 구현 예이다.

(1) x3DWorld world = new x3DWorld();
(2) world.setCamera(new x3DCamera(0,0,0);
(3) world.setLight(new X3DVector(0, 0, 1));
(4) x3DObject obj = new x3DSphere(150);
(5) obj.setColor(Color.gray);
(6) obj.translate(0, 0, 300);
(7) obj.rotate(2,10);
(8) world.render(obj, g);

(1) 가상의 3차원 좌표계 생성
(2) 카메라 생성과 3차원 좌표계에 설정
(3) 광원의 생성과 3차원 좌표계에 설정
(4) 3D Sphere 객체 생성
(5) 3D Sphere 색상지정
(6) z축으로 300 이동
(7) z축을 기준으로 10도 회전
(8) 3차원 좌표계에 3D 객체 렌더링

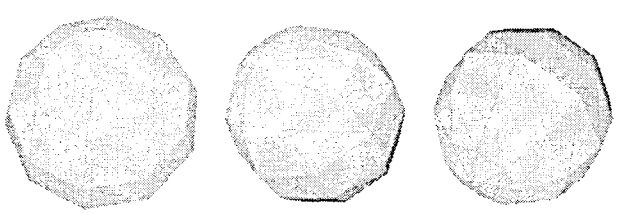

그림 7. 3D Sphere Object

V. 실 험

본 논문에서는 기술한 방법에 따라 실제로 Xlet과 3D Graphic Engine을 함께 전송하여 구동됨을 실험하였다. 또한, 데이터방송 표준간의 호환성을 검증하기 위해 상용 MHP STB과, 개발자용 OCAP STB에서 동일한 결과를 얻었다. ACAP STB에서도 정상적인 구동을 보여줄 것이라 사려된다. 특히, 본 논문은 3D Graphic Engine의 활용을 위해 T-Game을 위한 3D-Tetris를 제작하였다.

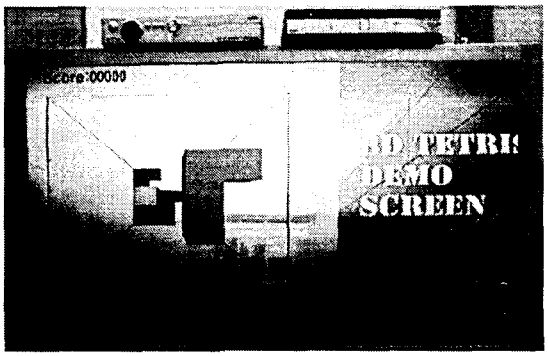

그림 8. 3D-Tetris demo

본 논문은 실험결과를 바탕으로 텍스처(texture) mapping, TV에 최적화된 렌더링, 캐스팅 방법 등에 관한 연구를 진행해 나가고자 한다.

참고문헌

[1] ETSI TS 102 812 V1.2.1 : Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1, 2003-06
[2] OC-SP-OCAP1.0-I13-041215, OCAP 1.0 Profile, 2004
[3] ATSC Candidate Standard: Advanced Common Application Platform (ACAP); Advanced Television Systems Committee, 2004-02
[4] Steven Morris, Anthony Smith-Chaigneau. "Interactive TV Standards."Focal Press. 2005.
[5] TR 101 202 Digital Video Broadcasting (DVB); Implementation guidelines for Data Broadcasting. 1999.
[6] BartCalder외, Java TV API Technical Overview : The JavaTV API Whitepaper ver 1.0, 2000
[7] ISO/IEC 13818-1 Generic Coding of Moving Picture and Associated Audio : Systems