Interactive TV를 위한 물리기반 Simulator Module 설계 및 구현

Design and Implementation of Physical-based Simulator Module for Interactive TV

김정환, JungHwan Kim*, 정문열, MoonRyul Jung**
*서강대학교 영상대학원 미디어공학과

요약 디지털방송은 애플리케이션(Xlet)을 이용해 보다 진보된 Interactive 방송프로그램을 시청자에게 전달 할 수 있다. Interactive TV 에서 애플리케이션은 MHP, OCAP, ACAP 등의 표준규약에서 정의된 API 를 기반으로 작성되며 MPEG-2 TS 를 통해 STB(Set-Top Box)에 전송되어 구동된다. 현재 이러한 애플리케이션은 고정된 이미지를 활용한 형태의 서비스가 주를 이루며 중력, 탄성력과 같이 실제와 유사한 움직임을 표현하거나 활용하는데 있어 미흡한 실정이다.

본 논문은 Interactive TV 에서 중요한 역할을 담당하는 애플리케이션(Xlet)을 위해, 현재의 STB 에서 가능한 물리기반 Simulator Module 을 설계하고 구현하였다. 이는 중력, 탄성력, 단진자운동과 같은 다양한 현상을 모형화(단순화)하고 수식화하여 적용한 것으로서, UI 와 Game 등에 그 활용도가 높을 것이다.

핵심어: 디지털방송, Interactive TV, Xlet, Physical-based Simulator Module

1. 서론

디지털방송은 MPEG-2 TS[1]를 통해 비디오/오디오와 더불어 이미지, 파일 등의 데이터전송이 가능하다는 특징이 있다. 이 데이터 영역에는 특별히 디지털방송에서 가용한 애플리케이션을 포함시켜 전송할 수 있는데 이는 Xlet[2]이라 불린다. Xlet은 필요할 때에만 셋톱박스(Set-top Box)에 전송되어 구동되는 애플리케이션으로 애플릿(applet)과 유사하다. 디지털방송에서 흔히 접할 수 있는 정보제공서비스와 게임 등이 모두 이에 해당된다. Interactive TV에서 이러한 애플리케이션(Xlet)은 시청자에게 매우 다채롭고 풍성한 서비스를 가능하게 해주는 중요한 요소이다.

현재 이러한 애플리케이션은 고정된 이미지와 선형적으로 단조로운 움직임을 이용한 서비스가 그 주류를 이루고 있다. 이는 낮은 사양으로 출시되는 셋톱박스, 비디오/오디오를 주 서비스로 삼고 있는 방송규약의 다소 미흡한 그래픽객체지원, 폐쇄적인 미들웨어 구조로 인한 Xlet의 RTOS(Real-Time Operating System) 접근불가 등이 큰 제한사항이기 때문이다. 일반적인 PC 환경에서는 그래픽객체만을 위한 높은 사양의 칩과 메모리가 존재하고 접근이 가능하며, OS에 최적화된 Win32API 같은 라이브러리가 지원된다. 하지만, MHP(Multimedia Home Platform)[2], OCAP(Open Cable Application Platform)[3], ACAP(Advanced Common Application Platform)[4] 등의 데이터방송 규약에 묶여 자바를 기반으로 하는 미들웨어가 탑재된 셋톱박스는 이러한 것들을 제공하기가 용이하지 않다. 또한, 출시되고 있는 셋톱박스는 32비트 운영체제인데 그래픽객체의 색상처리에 대해서는 유독 고정된 16, 18비트의 색상체계를 활용하고 있는 것들이 대부분이다. 이는 메모리의 이익을 취할 수는 있지만 고정된 비트라 그 활용이 어렵고, 부드러운 음영표현과 처리속도에 대해 제한사항을 가지게 된다. 즉, 현 상황에서 Xlet은 낮은 사양의 셋톱박스환경에서 고정된 색상만을 이용해 미들웨어에서 제공하는 API만을 사용해 무언가를 표현해야 한다는 것이다.

본 논문은 이러한 여러 제한사항을 가진 셋톱박스 환경에서 보다 진보된 Xlet 제작에 활용하기 위한 기법을 소개하고자 한다. 이를 위해 본고는 Moving Model, Spring Model, Gravity Model, Vibration Model을 정의하였고, 현 셋톱박스 환경에서 독자적으로 물리기반 운동을 표현할 수 있는 module을 설계하고 구현하였다.

본 논문에서 제안하는 기법은 UI, Game 등에 널리 활용되어 보다 진보되고 다채로운 데이터방송 서비스에 기여할 것이라 기대한다. 향후 고해상도의 HD 셋톱박스의 보급과 좀더 개방적인 구조의 미들웨어가 출시된다면 이러한 문제들은 자연스럽게 해결될 것이다.

2. Xlet 구동환경과 Life Cycle

본 장에서는 본 논문의 이해를 돕기 위해 Xlet의 구동환경과 Life Cycle에 대해 설명한다.

1. Xlet

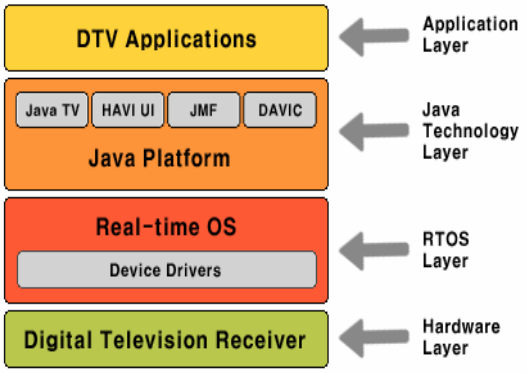
Xlet은 오브젝트카로셀(Object Carousel)[5]형태로 포장되어 MPEG-2 TS에 실려 셋톱박스로 전송되는 자바 애플리케이션이다. Xlet은 연동형, 독립형 애플리케이션으로 구분할 수 있다. 연동형 애플리케이션은 service에 종속적인 애플리케이션인데, 비디오와 정확한 동기화를 이루며 구동되는 Xlet은 동기화된 연동형 애플리케이션[6]이라 불린다. 연동형과 반대로 독립형은 특정 Service와 별도로 구동되는 애플리케이션을 말한다. 현재 국내의 대부분의 서비스(그림1)는 독립형으로 제작되어 방송된다.

<그림 1> Xlet application example

Xlet은 별도의 설치 과정 없이 다운로드 하고 바로 실행되고 소멸되는 라이프사이클(life cycle)을 가지고 있어 셋톱박스에 부담이 되지 않는다. 일반적으로 Xlet은 구동 당시에 다운로드 되어 작동하며 필요에 따라서는 로컬장치에 저장하여 사용되기도 한다. 또한 화면 처리(overlay)를 위한 기능들에 대한 규격이 정의되어 있기 때문에 TV출력에 적합한 형태를 가지고 있다. 더불어 Xlet은 데이터 방송 표준인 MHP[2], OCAP[3], ACAP[4]에서 공통으로 사용하고 있는 프로그램 구조로 데이터방송 미들웨어가 탑재된 시스템에서는 쉽게 구동이 가능하고, 애플리케이션매니저(application manager) 관리하에 있어, 프로그램들 사이의 화면 분할이나 자원(resource) 등에 대해 별도의 관리기능을 필요로 하지 않는다.

2. STB의 Xlet 구동환경

MPEG-2 TS[1]의 Data/Object carousel[5] 형태로 포장되어 전송되는 Xlet은 STB에 다운로드 되어 구동된다. Xlet이 STB에서 구동되기 위해서는 미들웨어의 API를 근간으로 하여 제작되어야 하고, 다운로드 된 Xlet은 미들웨어의 애플리케이션 매니저(application manager)에 의해 관장된다. 그림2는 셋톱박스에서 구동되는 Xlet을 위주로 그 구동환경을 도식화한 것이다.

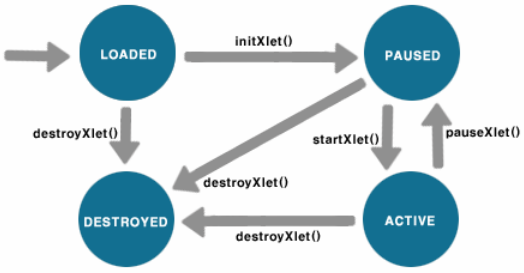


<그림 2> STB(Set-Top Box)의 Xlet 구동 환경

Hardware Layer는 STB의 하드웨어 시스템 자원을 일컫는 것으로서, Tuner, De-Multiplexer, Video/Audio Decoder 등이 대표적인 예이다. RTOS Layer는 STB에서 돌아가는 운영체제와 Device Driver가 위치한다. Java Technology Layer는 각각의 데이터방송표준에서 정의한 미들웨어 계층으로 채널전환, 비디오컨트롤, 시청자와 리모컨을 통한 이벤트처리 등을 위한 API를 제공해 준다. Application Layer는 Xlet이 존재하게 되는 계층이다.

3. Xlet 생명주기(Life Cycle)

애플리케이션매니저는 Xlet의 구동과 각종 예외상황, 생명주기, 다수의 Xlet에 대한 우선순위, 리소스 등을 관장하는 역할을 담당한다. Xlet은 크게 Loaded, Paused, Active, Destroyed의 4가지 상태(State)를 가지며, 애플리케이션 매니저는 initxlet(), startXlet(), pauseXlet(), destroyXlet() method를 통해 Xlet의 상태를 제어한다. 필요할때만 다운로드 되고 실행되며, 종료 할 때 모든 자원을 반납하고 소멸하는 Xlet은 그림 3과 같은 생명주기를 가진다.

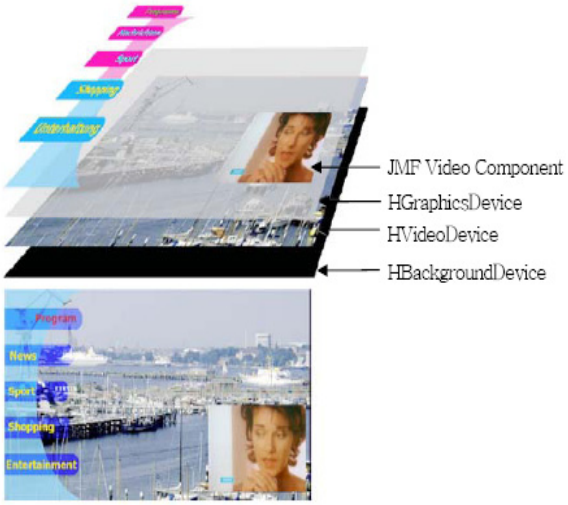


<그림 3> Xlet life cycle[2]

데이터방송은 이번 장에서 소개한 Xlet을 이용하여 T-Learning, T-Government, T-Banking등의 다양한 서비스 모델에 적용되어 활용된다.

4. Graphics Reference Model

MHP[2]는 Graphics Reference Model을 정의하고 있다. 이는 시청자에게 보여줄 화면구성에 대한 것으로, HAVi(Home Audio / Video Interoperability)[7]의 HBackgroundDevice, HVideoDevice, HGraphicsDevice에 대응되는 3개의 plane이 합성되어 시청자에게 전달된다. Background plane은 MPEG I-Fame[8], Video plane은 MPEG-2 video[8]를 그 주요 처리대상 삼으며, Graphics plane은 그래픽객체를 위한 plane이다.



<그림 4> Display planes [2]

방송 미들웨어에서 사용하는 그래픽 기능은 JavaTV [9]에서 정의되어 있는 내용을 기반으로 아주 간단한 기능만을 수행하며, alpha blending 기능이 추가되어 있다. 또한 JVM은 personal basic profile을 기반으로 하고 있기 때문에 Java에서 제공되는 기본 기능만을 수행한다. 즉 RMI, Swing과 같은 고급 기능은 사용할 수 없다. 이는 셋톱박스에서 지원할 수 있는 기능의 한계이다.

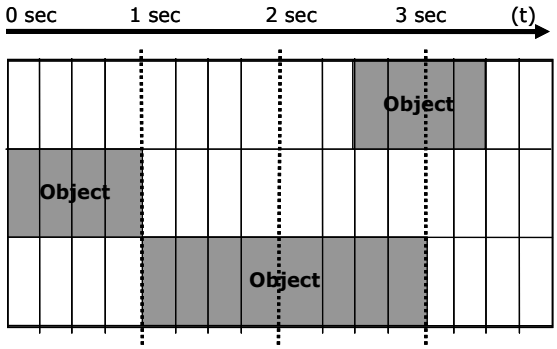
4. 물리기반 Simulation Model

본 장에서는 현 상용 STB에서 가능한 모델을 소개하고 이를 구현하기 위한 기법에 대해 논의한다 대표적인 모델은 Gravity, Vibration, Spring Model 등이다.

1. Virtual FPS Management Module

본 장에서는 다양한 물리현상을 STB에서 Simulation 하기 위한 VFMM(Virtual FPS Management Module)을 소개한다. 이는 Xlet의 화면 갱신 주기를 관장하며 자연스러운 애니메이션 효과를 표현하기 위한 필수 요소이다.

가상의 타임라인(virtual timeline)이란 쉽게 말해 일정시간에 몇 개의 단위 프레임을 구성하는지에 대한 것으로, 본 논문을 이를 통해 FPS를 정의하고 사용한다. 그림 5는 기본적인 구성 예이다.



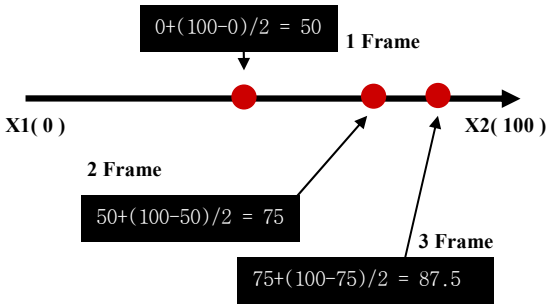
<그림 5> Virtual Timeline 구성 예

Virtual Timeline은 단일 오브젝트 또는 다수의 오브젝트를 관장하며 각각의 오브젝트가 애니메이션 되는 프레임수의 조정을 통해 시각적인 FPS를 조정할 수 있다. VFMM은 본 논문에서 제안하는 물리모델의 계산속도에 따라 적절한 FPS를 설정하도록 Xlet 프로그래머에게 위임하고 이를 기준으로 갱신하는 역할을 담당하게 된다.

2. Moving Model

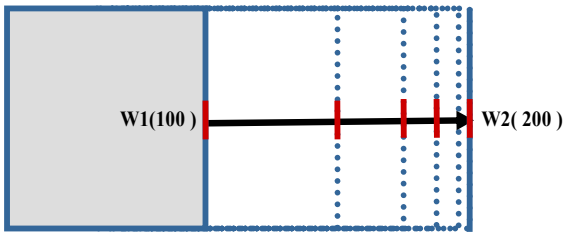
Moving Model이란 그래픽객체가 최종 목적지까지 이동할 때 적용되는 모델로서 계수를 통해 부드러운 움직임을 표현하기 위한 것이다. 초기 위치를 X1, 최종위치를 X2라 하면 X1의 이동거리는 X2-X1이다. 이때, X1이 이동거리(X2-X1)만큼 움직일 때 그 속도를 조절하면 다양한 효과를 표현할 수 있다. 그림 6은 어떤 물체가 X1을 출발하여 X2까지 움직일 때 부드러운 움직임을 표현한 것으로 계수(S)로서 2가 적용된 것이다.

식 1 : X1 = X1+(X2-X1)*S



<그림 6> Smooth Moving Example

그림6은 직선에 대해 부드럽게 멈추는 객체에 대해 표현한 것인데, 이를 객체의 크기로 간주한다면 부드럽게 확대되는 객체를 표현하는데 이용할 수 있다.



<그림 7> Smooth Model을 활용한 객체의 넓이표현

그림7은 Smooth Moving Model을 활용해 사각형 모양의 객체를 확대할 때 적용한 경우다. 매우 부드럽게 커지면서 확대되는 것을 볼 수 있다. 이 모델에서 중요한 변수는 계수인데 계수를 적절히 조율한다면 등속도, 등가속도운동, 마찰력 등을 다양하게 표현하는 것이 가능하다. 만일 색상에 적용한다면 제한된 색상의 범주에서 부드러운 색상변화를 표현할 수 있을 것이며, 곡선이나 원 등의 방정식을 이용하면 선형적인 움직임 이외에도 매우 다양한 표현이 가능 할 것이다.

3. Spring Model

탄성모델은 Moving Model을 기본으로 하여 발전된 탄력모델이다. Spring이 가진 성향을 표현하며, Hooke's law로 잘 알려진 용수철의 법칙과 Moving Model을 응용해 변형한 것이다. 용수철의 기본 원리가 원래 위치에서 많이 떨어져있으면 복원하려는 힘이 크고, 가까우면 작다라는 기본 개념에서 출발하였고, 프레임이 증가할수록 더욱 부드럽고 최종 목적지에 근사하게 접근하도록 표현하였다. 실제 공기저항이나 마찰력 등의 환경적 요인은 전혀 고려하지 않는다.

$$식\ 2 : X1 = a*(X1 - X2) + b*(X0 - X2) + X2$$

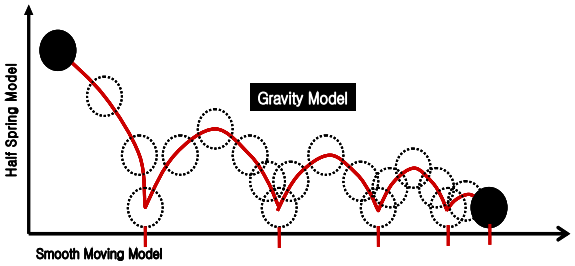
$$(a:-2<a<2,\ b:-1<b<0)$$

X0는 이전위치, X1은 현재 위치, X2는 이동할 최종위치를 나타낸다. Spring Model을 사용하면 객체의 탄성운동과 유사한 표현을 시청자에게 전달 할 수 있다. 객체의 크기, 위치, 색상 등의 다양한 곳에 활용이 가능하다

4. Gravity Model

Moving Model과 Spring Model을 적절히 활용하면 Gravity Model로서 활용할 수 있다. 임의의 객체가 지표면과 충돌했는지를 체크해 이를 변형해 주면 된다. 그림 8은 약간의 탄성력을 지는 공의 움직임을 표현한 것인데, 가로방향의 진행은 Moving Model로 공의 탄성과 중력의 움직임에 해당하는 세로방향은 Spring Model을 변형하여 적용하면 된다.

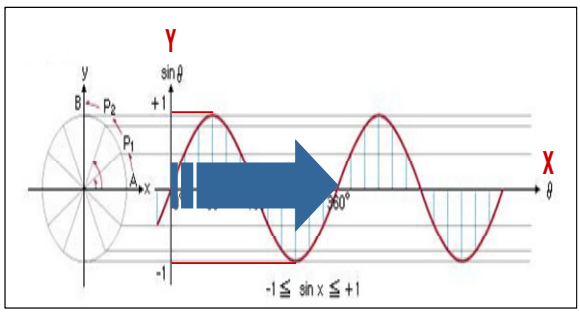
처음 공이 놓여질 X좌표를 Moving Model의 X1에 대입하고 모든 운동이 끝나고 지표면에 정지할 위치를 X2로 정의해 X의 Smooth Moving Model의 계수를 활용해 변량을 표현한다. 세로축(Y)은 공의 탄성을 표현하는데, 기본수식(식2)에서 살펴보면, 객체의 이전위치는 Y0, Y1은 현재위치, Y2는 최종적으로 객체가 충돌할 지표면의 위치이다. 객체가 바닥과 닿았을 경우 현재 위치(Y1)값을 고려해 바닥위로 올려주면 된다. 특히, 식2의 계수(a, b)를 적절히 조율하여 사용하면 객체의 속성을 더욱 잘 표현할 수 있으며 일종의 중력계수로서 활용된다.



<그림 8> Gravity Model을 이용한 객체의 움직임 예

5. Vibration Model

진동모델(Vibration Model)은 진동을 표현하는 모델로서 sign, cosign 곡선을 그 모태로 한다. 예를 들어 sign곡선의 세로축을 Y라 하면, Y는 초기값이 0이고, 360°를 기준으로 -1과 1사이에서 계속 진동하며 주기적인 값을 가지는 주기함수를 표현한다. 여기서 세로축에 대항하는 Y값에 적절한 계수를 곱하면 특정 범위에서 주기적으로 진동하는 값을 얻을 수 있다. 예를 들어 10을 곱하면 360°를 기준으로 -10과 10 사이에서 진동하는 주기적인 값을 얻을 수 있다. 이러한 Vibration Model을 Xlet에 적용하기 위해선 다행히 java.lang.Math 클래스가 지원하므로, 이를 이용해 라디안(radian)값을 얻으면 된다. 그림9는 이를 도식화하여 표현한 그림이다.



〈그림 9〉 Vibration Model System

그림 9와 같은 사인곡선을 활용한 진동모델을 사용하면 특정영역에서 진동하는 객체의 표현과 더불어 파도와 유사한 표현이 Xlet에서 가능해 진다. 파도의 움직임은 사인곡선과 같은 파형을 지니고 있기 때문이다. 또한 점을 이용해 부드러운 자유곡선을 그리는 것도 그리 어렵지 않게 구현 할 수 있다.

5. 실험 및 결론

본 논문은 기술한 방법에 따라 Xlet과 Simulator 모듈을 을 Object Carousel[2][5]로 MPEG-2 TS[1]에 실어서 셋톱박스에 전송한 후, OCAP에 정의된 Bound application[2] 형태로 실험하였다. 그림10은 실험환경의 전경이다.

<그림 10> OCAP 실험환경

그림 11은 본 논문에서 소개한 Gravity Model을 응용하여 만든 Snow Effect Simulation이다. 눈은 전혀 탄성이 없으므로 눈의 크기만을 중력계수로 적용하여 실험하였다. 배경은 MPEG-2 I-frame이며 눈송이 그래픽 객체이다. 300개로 가정하고 실험하였는데, 약 12~16FPS가 가능함을 확인하였다.

<그림 11> Snow Effect Simulation

그림12는 비가 창문을 타고 내려오는 것을 표현하기 위해 시도한 것이다. 빗방울은 gif 형식의 투명처리 된 이미지객체이며 배경은 MPEG-2 I-frame으로 제작된 것이다. 빗방울의 크기를 중력계수로 적용하였고 빗방울이 만나면 그 둘의 중력계수를 고려해 크기가 빠르게 확대 되도록 하였다. 또한, 빗방울이 내려오면서 남아있는 물 때문에 점점 커지게 되는데 이는 Smooth Moving Model을 적용하여 그 크기를 조율해 보았다. 약 30개의 빗방울로 실험하였고 8~12FPS가 적당한 비율로 보여졌다.

<그림 12> Rainy Window Simulation

본 논문은 현 상용STB에서 가능한 물리기반 운동을 단순화 하여 표현하기 위해 시도되었고, 제한된 환경에서 그 효과를 가장 잘 표현하기 위한 기법을 소개하고 있다. 본 논문은 단순한 계산만으로 하드웨어의 도움 없이 Xlet에서 가용한 API만을 이용하도록 제한하고 이를 실험하였다. 본 논문에서 제안하는 Simulation Model은 Xlet으로 제작되는 다양한 콘텐츠에 활용되고, 다양한 아이디어와 접목된다면 더욱 큰 가능성을 보여줄 것이라 기대한다.

참고문헌

[1] ISO/IEC 13818-1 Generic Coding of Moving Picture and Associated Audio : Systems

[2] ETSI TS 102 812 V1.2.1 : Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1, 2003-06.

[3] OC-SP-OCAP 1.0 : OpenCable™ Application Platform Specification OCAP 1.0 Profile. April, 2005.

[4] ATSC Standard: Advanced Common Application Platform(ACAP), Document A/101, 2, August, 2005.

[5] ISO/IEC 13818-6 Generic Coding of Moving Picture and Associated Audio : Digital Storage Media Command and Control. 1996.

[6] 정문열, 백두원. “연동형 데이터 방송 애플리케이션의 구조”. 방송공학회논문지. 제 9 권 제 1 호. 2004.

[7] HAVi, Inc. “Specification of the Home Audio/Video Interoperability (HAVi) Architecture, Chapter 8 – Level 2 User Interface,” version 1.1. 2001.

[8] ISO/IEC. 13818-2 Generic Coding of Moving Picture and Associated Audio: Video

[9] BartCalder 외, Java TV API Technical Overview : The JavaTV API Whitepaper ver 1.0, 2000.