분산 멀티미디어/하이퍼미디어 응용을 위한 MHEG 엔진 설계

이 세 훈[†] · 왕 창 종[††]

요 약

본 논문은 멀티미디어/하이퍼미디어 응용에서 MHEG 객체의 생성과 프리젠테이션 기능을 가지는 MHEG 엔진에 관한 연구이다. 설계된 MHEG 엔진은 전송된 MHEG 객체에 대한 디코딩후의 내부 포멧을 각각의 MHEG 객체가 가지는 객체 지향 방식의 표현 구조와 맵핑되는 구조로 정의함으로써, 디코딩이 용이해지고 인터프리터가 MHEG 객체를 쉽게 해석할 수 있다. 또한, 객체 정보를 계승과 포함 관계를 쉽게 표현할 수 있는 트리 형태로 관리함으로써, 외부 객체나 데이터 화일에 대한 동적인 참조 및 서브 객체에 대한 관리를 용이하게 할 수 있다. 프리젠테이션 동기화는 MHEG 복합 객체로부터 동기화 정보를 추출하여 시공간적 관계에 따른 이질적인 미디어들을 표현하고 제어할 수 있도록 동기화 모듈을 설계하고 내부 객체들의 메시지를 이용하여 동기화를 처리할 수 있는 알고리즘을 제안하였다. 본 논문에서 제안한 MHEG 엔진은 초고속 통신망에서 멀티미디어 응용 분야의 기반 기술이 될 수 있을 것이다.

Design of MHEG Engine for Distributed Multimedia/Hypermedia Applications

Sei-Hoon Lee[†] · Chang-Jong Wang[††]

ABSTRACT

In this paper, we design MHEG engine that can generate MHEG objects and present it to the users in Multimedia/Hypermedia Applications. In the MHEG engine, the transmitted MHEG objects decoded into internal format. For the easy interpretation of MHEG objects, we define internal format as to be matched for each MHEG object. We easily process object information using the tree data structure because object inheritance and possession can be represented in tree structure. Object inheritance and possession must be represented in the internal format because they used in resolving the reference to external object or data file. The presentation synchronization extracts the synchronization information from MHEG composite objects, representing and controlling heterogeneous media associated to spatio-temporal relation. In order to exactly represent the spatio-temporal synchronization included into MHEG composite object, we propose the algorithm that processes synchronization using the message of the synchronization module and the internal objects. MHEG engine proposed in this paper may be basic technology for multimedia application area using Korea New Net.

† 정 회 원:인하공업전문대학 전자계산기과
†† 정 회 원:인하대학교 전자계산공학과
논문접수:1995년 9월 20일, 심사완료:1996년 2월 8일

1. 서 론

멀티미디어란 단일한 디지탈 환경에서의 텍스트, 오디오, 비디오, 그래픽스 등의 여러가지 미디어가 디지탈 형태로 저장, 검색, 표현되며 하나의 통합환경 내에서 동시에 다루어지는 것을 의미한다. 이러한 멀티미디어 정보들은 네트워크상에서의 전송과 이기종간의 데이터 교환에 의해서가 아닌, 단지 지역적인 매체를 통한 저장과 실행을 통해서만 사용되고 있는 실정이다[1, 2].

초고속정보통신망 구축을 위한 계획 수립과 관련한 핵심 기술로 자리잡고 있는 멀티미디어의 응용은 교육, 광고, 오락, 출판, 원격 화상회의, 멀티미디어 전자우편, 원격 의료진단 등의 다양한 서비스를 다루고 있다. 이러한 서비스들은 일반적으로 실시간 처리 능력, 네트워크 환경에서의 사용, 다양한 시스템 사이의 정보 공유, 친숙한 그래픽 사용자 인터페이스(GUI)를 기본 요구사항으로 한다[3, 4].

네트워크 기반에서의 멀티미디어/하이퍼미디어 응용은 위에서 기술한 기본 요구사항을 다루어야 한다. 사용자는 이러한 응용을 통하여 원거리에서 멀티미디어/하이퍼미디어 데이터를 실시간으로 교환하고, 저작하며, 서버로의 접근과 자료 검색을 수행할 수 있어야 한다. 이러한 작업 수행을 위해서 멀티미디어/하이퍼미디어 정보의 표현과 데이터의 실시간 전송 형태에 대한 표준이 요구되며, 나아가 이기종 플랫폼상에서의 정보 공유를 위한 공통된 저장 형식의 설정도 필요하다.

현재 ISO/IEC JTC1/SC29/WG12 에서 표준화되고 있는 MHEG(Multimedia and Hypermedia information coding Expert Group)은 분산 환경에서 이기종 플랫폼상의 멀티미디어 응용들간에 최종 형태의 표현과 교환을 목적으로 멀티미디어/하이퍼미디어 정보의 코드화된 표현이다[5]. 이러한 MHEG을 기반으로 하는 응용 개발에는 MHEG 객체에 대한 생성, 전송, 해석, 처리 기능을 수행하는 MHEG 엔진의 설계가 필수적으로 요구된다[6, 7].

MHEG 표준에서는 엔진이 가져야할 기본적인 기능만을 서술하고 있으며, 구조는 정의하고 있지않다. MHEG 엔진에 대한 대표적인 연구로는 Umass Lowell [8], Mannheim[9]등과 MHEG 엔진을 내장한 응용으로는 HIRS[10], IMS1[11]등이 있다. 이들 연구에서의 MHEG 엔진은 객체 표현을 위한 프리젠테이션 측면을 중심적으로 기술하고 있으며, 실제적인 모듈의 상세 설계및 MHEG객체 생성을 위한 저작 측면을 고려하고 있지 않다.

본 논문에서는 MHEG 표준을 기반으로 멀티미디어/하이퍼미디어 응용의 핵심적 구성 모듈인 MHEG 엔진을 설계한다. 설계할 엔진은 MHEG 표준에서 제시한 기본적인 기능을 포함하며, 특히, 프리젠테이션을 위한 MHEG 객체의 해석을 위해 내부 표현 형식을 객체지향 접근방식으로 설계된 MHEG 객체의의 효율적인 맵핑을 위해 객체지향언어에서의 구조를 이용한다. 그리고, 객체의 계승과 포함을 나타내는 속성이 트리 구조의 특성과 유사하다는 점에서 내부 형식의 정보 관리를 위한 구조로서 트리 구조를 이용한다. 또한 MHEG 표준에서의 시공간적 동기화 정보를 분석하여 MHEG 런타임 객체 처리를 위한 MHEG 인터프리터 내의 동기화 모듈에서 사용되는 내부 객체 클래스를 설계한다. MHEG이 제공하는 동기화 정보를 처리하기 위해서 동기화 모듈에서 사용되는 내부 자료 구조와 MHEG 인터프리터의 액션 모듈과 링크 모듈간의 상호작용을 수행하는 동기화 모듈, 그리고 중심 루틴인 동기 메시지 처리 알고리즘을 제안한다.

멀티미디어/하이퍼미디어 응용에서 멀티미디어 문서의 최종적인 저장 모델과 실시간 전송을 위한 형태의 요구를 모두 충족시키는 MHEG 표준 기반의 MHEG 엔진 설계는 차후 네트워크를 통한 정보의 공유 측면을 고려할때 그 중요성을 들 수 있으며, 네트워크에 기반을 둔 새로운 응용에도 재사용될 수 있다는 점에서 본 연구의 의의가 있다.

2. MHEG 표준 및 엔진

본 장에서는 다양한 멀티미디어 표준중에서 실시간 대화형 멀티미디어 서비스를 목표로 설계된 MHEG 표준 및 현재 연구가 활발히 진행중인 MHEG 엔진을 분석한다. 또한 분석한 내용을 토대로 본 논문에서 설계할 MHEG 엔진의 구조와 기능을 정의한다.

2.1 데이터 상호 교환 모델

MHEG 표준은 분산 환경에서 이기종 플랫폼상의 멀티미디어 응용들간에 최종 형태의 표현과 교환을 목적으로 멀티미디어 정보의 코드화된 표현을 정의한다. 이를 위해 MHEG 표준은 상호작용 및 멀티미디어 동기화, 실시간 표현, 실시간 상호교환, 최종형태의 표현등을 그 범주로 두고 있으며, 특히 통신망을 통한 실시간 정보전송 및 교환에 그 촛점을 맞추고 있다[12, 13].

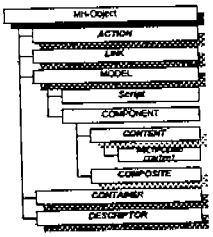
두 응용사이의 데이터 교환은 응용이 다루는 정보의 상위 단계 표현이 공유를 목적으로 하지 않는 단일 미디어 형태로 이루어 지는 경우와 같은 정보를 공유할 목적으로 교환하는 두 가지 경우로 나눌 수 있다. 전자는 MPEG, JPEG 등과 같이 데이터 압축을 통하여 응용 영역과는 독립적으로 단일미디어 형태로 교환되는 경우이다. 후자는 데이터 정보에 대한 구조화된 표현 구조를 지원하여 데이터 자체가 아닌 데이터의 표현 정보만을 국제 표준 언어 또는 객체를 통하여 상호 교환되는 경우이다. 표준 언어의 형태로는 C, C++, SGML, HyTime과 스크립트 언어인 Hypertalk, ToolBook, ScriptX 등이 있으며, 객체 유형으로는 MHEG이 있다[7].

MHEG을 기반으로 응용될 수 있는 4가지 모드로는 멀티미디어 문서의 생성과 편집시의 최종적인 저장 모델, CD형태처럼 사용자에게 최종형태 디지탈 매체의 전송을 위한 포맷, 교육, 훈련, information-on demand 등 네트워크를 통해 서버로부터 클라이언트로의 실시간 전송을 위한 포맷, 내부 응용 사이의 데이터 교환을 위한 포맷등이 있다[5, 6].

표준을 위해 객체지향방식의 접근을 수용하는 MHEG에서 멀티미디어 정보의 코드화된 표현을 MHEG 객체라 부르고, 객체 클래스는 일관성 있는 특정한 구조를 갖는 멀티미디어 객체들의 집합으로 정의하고 있다[9].

(그림 1)은 위의 기술 내용을 담고 있는 객체 클래스들의 관계를 상속 트리로 표현한 것이다.

MH 객체 클래스는 가장 상위에 있는 클래스로서 모든 서브 클래스가 공통적으로 갖는 애트리뷰트들을 갖는다. 여기에는 MHEG임을 가리키는 표준 식별자, MHEG의 버젼을 나타내는 표준 버젼, 어느 MHEG 클래스에 속하는지를 나타내는 클래스 식별자, 교환되는 MHEG 객체를 분별하기 위한 MHEG



(그림 1) MHEG 클래스 상속 트리
(Fig. 1) MHEG class inheritance tree

식별자, 교환되는 MHEG 객체에 대해 보다 자세한 해설을 나타내는 애트리뷰트가 있다.

액션 클래스는 MHEG 객체와 런타임 객체, 그리고 채널의 초기 행위를 결정하는데 사용하는 클래스로서 객체 내의 코드 실행을 활성화시킨다. 이러한 액션 클래스의 인스턴스인 액션 객체들은 링크 효과를 기술하기 위하여 링크 객체내에서 사용된다.

링크 클래스는 공간적, 시간적, 그리고 조건 관계와 MHEG객체, 런타임 객체, 채널에 대한 연결을 나타내기 위해서 정의된다. 링크 객체는 하나의 출발점 객체로 부터 하나 이상의 목적지 객체들 사이의 관계를 나타내는 기능을 가지며, 링크의 개시를 위한 트리거 조건과 부수적인 조건, 그리고 조건이 만족됐을 때의 링크 효과를 나타내는 애트리뷰트를 가지고 있다.

내용 클래스는 내용 프리젠테이션을 위해 요구되는 정보를 포함하는 파라미터의 집합을 가지며, 미디어 정보의 코드화된 표현을 포함하거나 참조하기 위한 클래스이다. 미디어 데이터는 JPEG, MPEG과 같은 국제 표준에 따라 부호화된다. 파라미터 집합에는 내용 데이터, 데이터의 인코딩 설명을 위한 후크(hook), 내용의 크기를 나타내는 애트리뷰트와 내용의 지속시간을 나타내는 애트리뷰트로 구성된다.

다중 내용 클래스는 내용 클래스의 서브 클래스로 여러가지 멀티미디어 데이터의 코드화된 표현을 포함하거나 참조하는 것과 관련한 애트리뷰트를 정의한다. 이 클래스는 포함하거나 참조하는 스트림(stream) 즉, 내용 데이터의 유일한 식별자와 타입 그리고 지원해야 할 프로그램을 기술하는 구조를 지원한다.

복합 클래스는 멀티미디어/하이퍼미디어 정보의 교환을 위한 구조와 객체들간에 동기화를 표현하는 구조를 갖는 클래스이다. 크기, 속도, 볼륨, 선택 스타

일에 대한 내용의 프리젠테이션과 다양한 표현 대상 사이의 시공간적인 관계와 특정 조건에 따라 수행되는 링크, 그리고 링크에 의해 수행되는 액션에 대한 반응 행위를 기술하는 애트리뷰트를 가진다. 복합 객체는 내용 객체와 복합 객체를 포함할 수 있으며, 조건에 따라 객체에 대한 행위를 발생시킬수 있는 링크 객체, 링크 효과에 의해 시공간적인 동기화 표현과 관련된 액션 객체를 포함할 수 있다. 이것은 화면 단위의 프리젠테이션을 지원해야 하는 경우 기본 단위로 사용할 수 있는 구조이며, 전송시 여러 객체를 묶는 컨테이너로 사용 가능하다.

2.2 기존 MHEG 엔진 분석

MHEG객체로 표현된 M/H 정보는 MHEG 엔진을 통해 인코딩/디코딩 및 프리젠테이션 처리를 위한 객체 해석, 그리고 사용자와의 상호 작용 등을 하며, MHEG 표준에 기반한 어떠한 응용도 엔진을 필요로 한다.

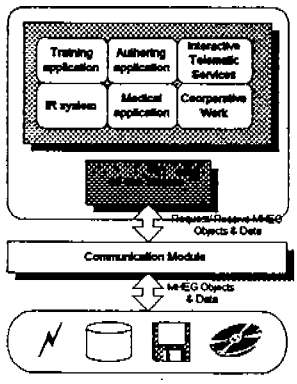
미국의 메사츄세소 대학(University of Massachusetts)의 MHEG 엔진[8]은 스케쥴러, 프리젠테이션 모듈, 객체 디코더로 구성된다. 엔진은 플랫폼상에서의 프리젠테이션 서비스와 인터페이스하며, 객체의 상태가 준비됐을때 객체의 표현을 요청한다. 사용자와의 상호작용은 사용자가 개입한 객체의 상태 변화를 야기시키는 프리젠테이션 시스템으로 부터 엔진으로의 이벤트를 발생시키며, 이러한 상태 변화는 엔진내의 스케쥴링에 중요한 요소가 된다. 이 엔진은 프리젠테이션에서의 구성과 상호작용을 정의하는 객체들을 제어하는데 주요 기능을 하며 엔진에 대한 제어는 객체가 검색되고 프리젠테이션되는 시기를 결정하는 스케쥴러내에 포함된다. 또한, 스케쥴러는 프리젠테이션을 운용하기 위하여 현재 활성화된 객체와 링크 객체의 상태 변화에 의존적이다. 그러나, 메사츄세스 대학의 MHEG 엔진은 MHEG 객체의 생성 기능이 없고, 이러한 이유로 엔진을 위한 테스트 작업은 프리젠테이션 측면에서만 진행되고 있다.

독일의 Mannheim 대학의 MHEG 엔진[9]은 3개의 모듈과 네트워크에 대한 접근과 디코딩을 수행하는 TCP 기반의 MRP(MHEG request/response protocol) 클라이언트로 구성된다.

엔진의 동작은 응용에 의해서 초기화되며, MRP 클라이언트에 의해 요청된 MHEG 객체가 전송 및 디코드되고, 이 객체를 MHEG 엔진에 보냄으로써 프리젠테이션이 시작된다. 객체 관리자는 실행 사이클 동안 엔진속으로 로드된 모든 MHEG 객체를 저장하는 리소스이며, 객체가 또 다른 객체에 대한 참조를 포함하면 이러한 객체에 대한 전송을 MRP 클라이언트에 요청한다. 인터프리터는 MHEG 엔진의 중심 부분으로 이벤트-드라이븐(event-driven) 방식을 취하며, 이벤트들은 링크 프로세서와 사용자 인터페이스를 통해서 들어오고 메시지 큐(message queue)에서 관리된다. 이벤트들은 인터프리터에 의해 해석되고, 프리젠테이션 서비스를 통하여 사용자 화면에 보여지게 된다. 그러나, Mannheim 대학의 MHEG 엔진은 MHEG 객체의 해석을 위한 디코더를 통신 모듈의 요소로 구성하고 있다. 엔진내에 디코더를 포함하는 것에 비해 이러한 구성은 디코딩 작업 후 지역 문법으로 변환된 MHEG 객체의 해석을 위해 통신 모듈상에서 엔진속으로 로드해야 하는 부담이 따르게 된다.

2.3 MHEG을 기반한 응용

본 절에서는 MHEG을 기반으로 한 응용으로써 IMS1, HIRS에 대해서 알아본다. MHEG을 기반으로 하는 응용은 매우 광범위하며, (그림 2)와 같은 공통적인 구조를 가지고 있다.



(그림 2) MHEG 기반의 다양한 응용
(Fig. 2) MHEG-based Applications

IMS1(Integrated Multimedia Services at about 1 Mbit/s)[11]은 공공 네트워크를 통해 VOD(Video On

Demand)등을 포함한 일반적인 멀티미디어 검색 서비스의 가능성을 실험하기 위해 개발되었다. 현재 연구 목적의 프로토타입시스템 개발이 완료되었고, 상업용 응용 개발이 시작되었다. IMS1에서는 상호 작용적인 실시간 검색만을 지원한다.

IMS1에서는 M/H 데이터를 표현하기 위해 MHEG 표준을 사용하였다. 현재 IMS1은 주문형 비디오와 같은 실시간 상호작용적인 서비스만을 제공한다. 그리고 IMS1에서는 클라이언트의 처리 능력을 최대로 하고, 클라이언트와 서버 사이의 네트워크 통신량을 최소로 하기 위한 클라이언트/서버 유형을 제안하였다.

ETRI에서 설계한 HIRS(Hypermedia Information Retrieval System)[10]는 정보 공유에 촛점을 두고 데이터 표현과 처리를 위해 MHEG을 기반으로 한 클라이언트/서버 구조의 정보 검색 시스템이다. HIRS에서 MHEG으로 코드화된 정보는 원거리의 서버에 저장되며, 다수의 클라이언트가 N-ISDN을 통하여 공유할 수 있는 구조를 가지고 있다. 클라이언트 시스템은 사용자와 상호작용하며, 사용자로 부터 데이터를 입력받거나 요구된 데이터를 서버 시스템으로 요청하는 역할을 수행한다. 서버 시스템은 클라이언트 시스템으로 부터 요청을 받고, 저장된 객체로 부터 요청된 데이터를 검색하고 전송하는 역할을 수행한다. HIRS의 클라이언트 시스템에 포함된 MHEG 파서와 포멧터는 각각 MHEG 객체의 인코딩 및 디코딩 역할을 수행하며, MHEG 인터프리터에서 내부 자료구조로 변환된 객체의 처리를 담당하게 된다. MHEG 인터프리터는 내부 자료구조로 변환된 MHEG 객체를 해석하여 다양한 액션을 수행한다. 특히, 다른 엔진과는 다르게 내부 자료구조로 변환된 MHEG 객체를 관리하기 위하여 엔트리(entry) 테이블을 포함하고 있다. 그러나, MHEG 엔진을 기반으로 한 하이퍼미디어 정보 검색 시스템의 프로토타입 시스템 구현 환경이 멀티태스킹을 지원하지 않는 비선점(non-preemption)형의 MS-Window 3.1 운영체제에서 개발되었기 때문에 MHEG 모델에서 다루는 MHEG 이벤트들의 동시성 요구 사항과 멀티미디어 데이터의 실시간 제약 조건을 만족시킬 수 없는 단점이 있다.

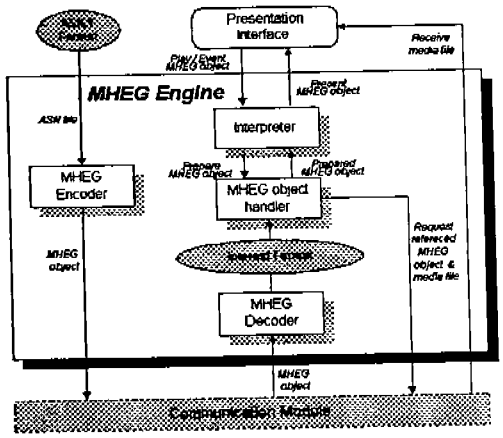
3. 객체의 효율적 처리를 위한 MHEG엔진

본 장에서는 기존 연구에서 제시하고 있는 MHEG 엔진을 분석하여 분산 M/H 응용 개발 환경에 적합한 엔진을 제안한다. 설계한 MHEG 엔진은 MHEG 객체의 생성, 전송, 변환, 해석, 처리가 가능한 구조를 가지며, 특별한 응용에 재사용될 수 있도록 독립적으로 모듈 설계를 한다.

3.1 MHEG 엔진 구조

본 논문에서는 엔진의 세부 모듈 설계 및 자료 구조를 정의하며, 이를 기반으로 한 응용, 그리고 데이터베이스나 네트워크를 통한 유기적인 상호 동작이 가능할 수 있도록 엔진을 구축한다. 또한, 시스템 혹은 매체를 통해 교환 가능한 MHEG 객체를 생성할 수 있는 인코더를 추가한다. 각 모듈은 고유의 기능을 위해 독립적으로 설계하며, 모듈 사이의 통신은 메시지 패싱에 의해 유기적으로 상호 동작하는 구조를 가진다. 그리고 효율적인 MHEG 객체의 해석을 위해 클래스 형태의 내부 형식을 구축하고, 로드된 MHEG 객체 정보는 MHEG 객체의 상속성을 잘 표현할 수 있는 트리를 이용하여 관리하기 위한 자료구조와 알고리즘을 제안한다. 또한, 표준에 정의되어 있지 않은 미디어 화일의 소스 참조 방식을 위한 메커니즘을 정의한다.

(그림 3)은 설계한 MHEG 엔진의 구성 모듈과 객체의 흐름을 나타낸 구성도이다. MHEG 엔진은 네개의 모듈, 즉, MHEG 인코더, MHEG 디코더, MHEG



(그림 3) MHEG 엔진 구성도
(Fig. 3) Configuration of MHEG Engine

객체 관리기, 인터프리터 모듈로 구성된다.

MHEG 인코더는 ASN 화일을 입력으로 받아 ASN.1(Abstract Syntax Notation One) 인코딩 규칙에 의해 MHEG 객체를 생성하는 역할을 하고, MHEG 디코더는 다양한 매체를 통해 로드된 MHEG 객체의 디코딩 기능 및 내부 형식으로의 변환 기능을 수행한다. MHEG 객체 관리기(object handler)는 내부 형식으로 변환된 MHEG 객체에 대한 관리 및 프리젠테이션을 위한 객체를 준비하는 역할을 한다. 객체에 대한 준비는 인터프리터로부터 준비(Prepare MHEG object) 메시지가 들어올 때 시작되며, 준비 메시지는 준비할 MHEG 객체의 식별자에 대한 포인터를 포함한다. 객체에 대한 준비 작업은 참조를 요구하는 MHEG 객체나 미디어 화일들에 대한 로드이며, 로드가 되어 MHEG 객체 관리기에 등록됐을 때, 준비가 끝났음을 알리는 메시지(Prepared MHEG object)를 인터프리터로 보낸다. 이 메시지 또한 준비된 MHEG 객체의 식별자에 대한 포인터를 포함한다. 인터프리터는 MHEG 객체 관리기를 통하여 준비된 객체로의 접근, 해석 및 동기화된 표현 기능 그리고 사용자로부터의 이벤트를 받아 처리하는 기능을 수행한다.

MHEG 표준에서는 MHEG 객체를 표현하기 위한 형식으로 ASN.1, SGML, 그 외의 방법등을 기술하고 있으며, 기본적인 방식으로는 ASN.1(Abstract Syntax Notation.1 ISO 8824) 표기법[14]을 사용하고 있다. 따라서, 제안 모델에서는 MHEG 객체 표현 형식으로 ASN.1 표기법을 선택하였다. ASN.1 표기법은 MHEG 표준에서 모든 데이터 구조를 형식화하여 표현하는 수단이다. ASN.1 표기법으로 생성된 MHEG 객체는 전송을 위한 문법인 ASN.1 인코딩 규칙(Basic Encoding Rules for ASN.1 ISO 8825) [15]에 의해 전송 형태로 변환된다.

3.2 MHEG 인코더/디코더

본 절에서는 MHEG 객체를 전송 형태로 변환시키는 인코더와 수신된 MHEG 객체의 프리젠테이션을 위하여 내부 형식으로 변환하는 디코더를 설계한다. 디코더를 통해 변환된 내부 형식은 C++ 언어의 클래스 구조로써 MHEG 객체 생성시의 구조와 유사한 객체 지향적인 형태이다. 따라서, 객체 처리시 새로운 함수를 생성하지 않고 클래스 구조내의 멤버 함수를 이용 메시지 패싱 방식에 의해 처리되며, 새로운 응용에서의 재사용성이 고려된다.

MHEG 인코더와 디코더를 구축하기 위해서 먼저 ASN.1 표기법에 의해 생성된 MHEG 객체 표현을 C++ 클래스 형태로 형성하는 ASN 클래스 생성기를 설계하였다. ASN 클래스 생성기는 ASN.1 정의를 C++ 클래스로 변환시켜 주는 프로그램이다. ASN 클래스 생성기를 통하여 출력으로 얻어진 C++ 클래스는 MHEG 인코더에 의해 전송 가능한 형태인 바이트 스트림으로 변환되며, 이것은 프리젠테이션시 디코더에 의해 내부 형식으로 변환된다. ASN 클래스 생성기는 ASN.1 표기에서 정의된 데이터를 C++ 멤버 함수를 포함한 클래스로 생성하며, 자체적으로 인코딩 및 디코딩이 가능한 구조가 된다. 이러한 구조는 MHEG 인코더와 디코더의 역할을 수행할 수 있는 멤버 함수를 포함하며, 내부 자료구조로의 변환 및 정보 접근에 대한 용이성을 제공한다는 장점이 있다.

MHEG 인코더는 객체에 대한 실제 데이터 정보를 입력으로 받아 객체의 타입에 따라 각각 전송 문법으로 변환한다. 이때, M/H 객체 정보에 대한 값들, 즉 MHEG 객체 클래스의 인스턴스인 MHEG 객체는 미리 선언된 C++ 형태의 MHEG 객체 클래스 내의 애트리뷰트에 대입되며, 인코드 함수를 통하여 각 클래스 단위로 인코딩된다. 인코딩된 데이터는 바이트 스트림 형태로 이진값을 가진다.

MHEG 디코더는 ASN.1 코딩 규칙에 따라 표현된 MHEG 객체에 대한 디코딩 즉, 내부 형식으로의 변환 기능을 수행한다. 본 논문에서 디코더는 MHEG 객체를 내부 형식으로 1대1 매핑하여, 디코딩할 수 있는 디코더를 설계하였다. 이것은 MHEG이 각 멀티미디어 객체의 속성을 객체 지향적인 방식으로 모델링하므로써 이를 프리젠테이션시에도 동일한 형태의 객체를 사용하여 모델링할 수 있다는 점에 착안하였다. 먼저 MHEG 디코더는 바이트 스트림 형태의 MHEG 객체를 디코드 함수를 이용하여 C++ 형태의 구조로 변환시킨다.

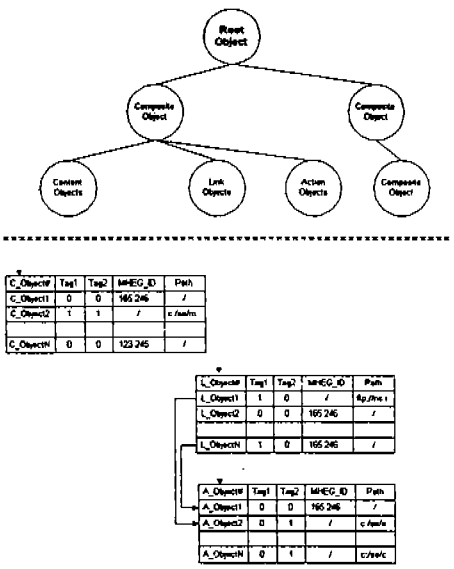
3.3 MHEG 객체 관리기

기존 연구에서는 엔진이 다루는 MHEG 객체의 관리를 위한 기능이 결여되어 있다. M/H 응용의 요구에 따라 프리젠테이션되는 MHEG 객체는 순차적 또

는 병행적으로 처리될 수 있다. 이러한 처리를 위해서는 엔진 내부적으로 MHEG 객체를 저장 관리하고 필요에 따라 로드할 수 있는 기능을 가지는 요소가 필수적이다.

제안 엔진에서는 내부 형식으로 변환된 MHEG 객체를 관리하며, 인터프리터에 의해 MHEG 객체가 사용될 수 있도록 준비하고 내용 데이터의 접근에 대한 정보를 관리하는 객체 관리기를 포함한다. 객체 관리기는 자신의 국부(local) 시스템 영역과 네트워크를 통해 연결된 원거리의 서버(server) 시스템 영역에 존재하는 MHEG 객체를 처리 대상으로 한다. 영역내 MHEG 객체의 관리를 위한 구조로 MHEG 객체가 가지는 상속성을 쉽게 표현할 수 있는 트리를 사용하며, 참조를 위한 구조로 테이블을 이용하였다. 트리의 구축은 엔진내로 MHEG 객체가 로드되고 내부 형식으로 변환됐을 때 시작되며, 실시간 프리젠테이션을 위해 트리의 깊이(depth)가 3까지로 제한하여 구성하였다.

(그림 4)는 내부 형식으로 변환된 MHEG 객체에 대한 정보를 추출하여 형성된 참조를 위한 MHEG 객체 트리이다. 이 경우 디폴트로 로드된 루트 객체가 먼저 준비되고, 루트 객체가 포함하고 있는 객체

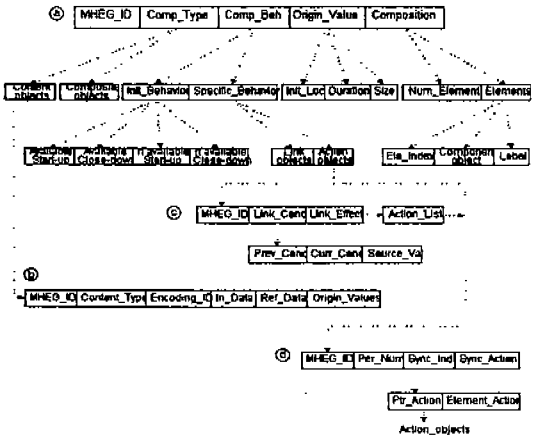


(그림 4) 로드된 객체의 관리를 위한 MHEG 객체 트리 구조
(Fig. 4) MHEG object tree for managing loaded objects

혹은 참조를 필요로 하는 객체들은 트리 구조의 노드나 터미널(terminal)을 구성하게 된다. 노드는 상위 객체가 포함하고 있는 객체들에 대한 정보를 가지고 있으며, 터미널은 객체 자신의 정보나 내용 객체의 서브 내용들에 대한 정보로 구성된다. 트리에서 터미널은 객체에 대한 참조 정보를 가지는 테이블에 대한 포인터 값을 가지며, 테이블로 구축된 참조 정보는 각 객체에 대한 인덱스 번호, 기존의 참조 여부를 나타내는 태그(Tag1), 내부 혹은 외부 참조 여부를 나타내는 태그(Tag2), 객체에 대한 위치 정보를 가지는 MHEG 식별자(MHEG_ID), 참조 경로명으로 구성된다.

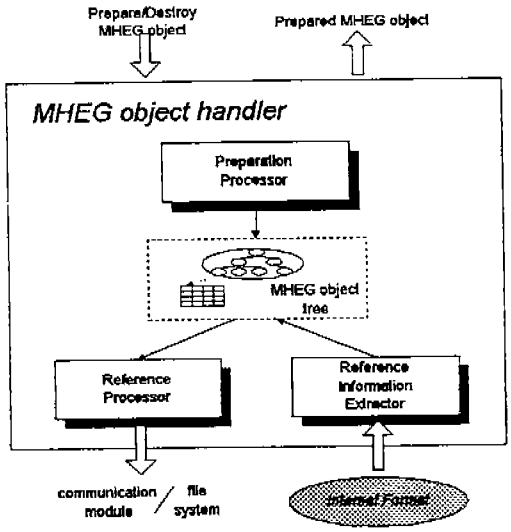
(그림 5)는 내부 형식으로 변환된 MHEG 객체의 구조이다. 내부 형식으로 변환되는 객체는 복합 객체(ⓐ), 내용 객체(ⓑ), 링크 객체(ⓒ), 액션 객체(ⓓ) 이다.

MHEG 객체 관리기는 내부 형식으로 변환된 MHEG 객체를 관리하며, 인터프리터에 의해 MHEG 객체가 사용될 수 있도록 준비하고 내용 데이터의 접근에 대한 정보를 관리하는 기능을 수행한다. 또한, 로드된 MHEG 객체가 다른 객체에 대한 참조를 포함하면, 객체에 대한 전송을 통신 모듈에게 요청한다.



(그림 5) 내부 형식으로 변환된 MHEG 객체의 구조
(Fig. 5) MHEG object structure converted to internal form

(그림 6)은 MHEG 객체 관리기의 전반적인 기능을 나타내는 모듈 구성도이다.



(그림 6) MHEG 객체 관리기의 모듈 구성도
(Fig. 6) Module configuration of MHEG object handler

요청된 객체의 준비(prepare)와 프리젠테이션된 객체의 반납(destroy)을 위한 액션은 인터프리터에 의해 시작된다. 인터프리터에서 MHEG 객체의 준비 메시지가 들어왔을 때, MHEG 객체 관리기내의 준비 처리기(Preparation Processor) 모듈은 해당 객체에 대한 MHEG 식별자를 조사한다. 해당 객체에 대한 조사는 먼저 MHEG 객체 관리기가 관리하는 MHEG 객체 트리에서 수행되며, 해당 객체가 존재하지 않는 경우는 참조 처리기로 해당 객체에 대한 참조 요청을 수행한다. 해당 객체가 준비됐을 때 즉, 내부 형식으로 변환된 MHEG 객체에 대한 준비 완료 메시지는 인덱스 번호를 식별자로 하는 해당 객체에 대한 포인터를 포함한다. 해당 객체에 대한 반납은 MHEG 객체 트리내에서 이루어지며 준비 처리와 마찬가지로 인터프리터로 부터 반납 메시지가 들어올 때 수행되며, MHEG 식별자를 인덱스로 조사하여 해당 객체를 삭제한다. 인터프리터의 이러한 액션을 통하여 객체 관리기내의 MHEG 객체에 대한 새로운 트리의 구성과 트리 해제 처리가 수행된다.

참조 처리기(Reference Processor)는 객체 트리내에 해당 객체가 존재하지 않는 경우 객체의 외부 경로명을 이용하여 통신 모듈이나 화일 시스템으로 객체 요청에 대한 메시지를 전달한다. 네크워크를 통하여 수신된 MHEG 객체는 다시 디코더에 의한 디코딩을 거친 후 수신된 객체를 포함하는 객체의 자노드로 위치된다. 상위 객체에 포함되는 객체의 참조 및 로드는 로드된 객체를 기준으로 트리의 깊이가 3까지 수행된다.

MHEG 객체 관리기는 내부 형식의 MHEG 객체 내에서 사용되는 엔터티 즉, MHEG 객체나 미디어 화일의 참조를 위해 [알고리즘 1]과 같은 참조 처리 과정을 수행한다.

[알고리즘 1] MHEG 객체 내의 엔터티 참조 알고리즘

```
MHEG_ObjectRef(MHEG_Object)
{
Access_Status = Get_Status(MHEG_Object);
// MHEG 객체로부터 엔터티참조에 대한 전후순서 파악
Ref_Status = Get_RefStatus(MHEG_Object);
// 기존의 참조 여부를 조사
if (Access_Status = 1) {//즉시, 참조 기능을 수행
  if (Ref_Status != 0) {
    Access();//통신모듈을 통한 접근(객체, 미디어)
    if (Access error) errorhandling();
  }
  else {
     Reference = Get_RefInform(MHEG_Object);
      // 외부 or 내부 참조 여부를 파악
     if (Reference = 1) {// 외부 참조
      Access(system_id or public_id);
      // system_id또는 public_id 를 통한 접근
      if (Access error) errorhandling();
     }
  else if(SearchPath(preparation path) = NULL){
      // 내부 참조
        Access(MHEG-ID);
      // MHEG_ID를 통한 접근
        if (Access error) errorhandling();
      }
      else errorhandling();
  }
}
else if (Ref_Status != 0) return;
      else Waiting;
}
```

[알고리즘 1]에서 MHEG 객체 관리기가 참조를 필요로 하는 상황은 두가지이다. 첫번째는 요구되는 엔티티를 즉시 참조할 필요가 없는 경우이고, 두번째는 즉시 참조해야 하는 경우이다. 첫번째의 경우에 해당 엔티티에 대한 참조가 기존에 있었는지 확인하고, 그렇지 않은 경우에는 전방 참조를 통하여 엔티티에 대한 외부 또는 내부 참조를 위한 정보를 갖게 된다. 이 경우 MHEG 객체 관리기는 엔티티가 요구될 때 참조 정보를 가지고 참조에 대한 처리를 할 수 있다. 두번째 경우에는 MHEG 엔진에 의해서 바로 통신 모듈을 통하여 엔티티에 대한 접근 및 로딩을 수행할 수 있는 경우와 외부 혹은 내부 참조 메커니즘을 통하여 엔티티 접근을 수행하는 경우로 나눌 수 있다. 후자의 경우에는 해당 엔티티에 대한 public_id 또는 system_id로 기술된 경로 정보를 통하여 처리하게 된다. public_id(예 : //ftp : /nms.inha.ac.kr/pub/selab/MHEG /campus.jpg)는 SGML에서 정의한 문법에 따르는 문자열로써 일종의 URL (Uniform Resource Location)과 같은 형태이다. system_id(예 : c : \selab\MHEG\ campus.jpg)는 시스템내의 특정 정보를 명시하기 위한 수단으로 사용되는 시스템 의존적인 참조 방식으로 해당 엔티티에 대한 경로에 따라 엔티티로 접근한다. 내부 참조는 엔티티내에 코드화된 참조 정보 즉, MHEG 식별자를 통하여 참조를 해결한다.

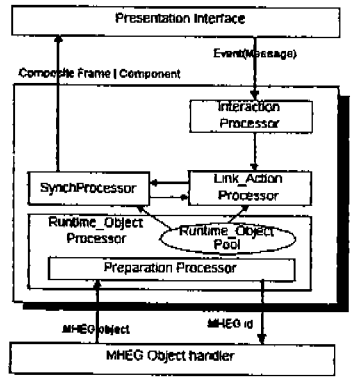
3.4 인터프리터

인터프리터는 MHEG객체를 해석하여 프리젠테이션 인터페이스에 적절한 정보를 제공하는 기능을 수행한다. 이러한 기능에는 MHEG의 액션과 링크객체의 처리 및 사용자 상호작용 처리와 MHEG 내용 객체 처리를 위하여 MHEG 복합 객체로부터 추출된 동기화 정보를 이용한 동기 처리의 역할이 MHEG 인터프리터에서 수행하는 기본적인 기능이다. 본 절에서는 이러한 기능을 수행하는 인터프리터를 설계하며 MHEG엔진의 다른 모듈과의 인터페이스를 정의한다.

가. 인터프리터 구성도

인터프리터의 기능을 분리하면 사용자 상호작용 처리와 MHEG 내용 객체의 동기 처리, 그리고 MHEG의 링크와 액션 처리를 위한 부분으로 나누어질 수 있다.

본 논문에서는 크게 4개의 모듈로 인터프리터를 구성하였다. 사용자와의 상호작용을 처리하는 상호작용 처리기, 동기를 처리해 주는 동기 처리기, MHEG 객체의 액션과 링크를 처리하는 링크액션 처리기와 런타임 객체의 생성 및 준비 과정을 처리하는 런타임 객체 처리기로 구성되어 있다. 인터프리터의 구성도는 (그림 7)과 같다.



(그림 7) 인터프리터의 구성도
(Fig. 7) Configuration of MHEG interpreter

본 논문에서 제안한 인터프리터는 MHEG엔진의 MHEG객체관리기와 응용프로그램 수준인 프리젠테이션 인터페이스와 각각 공동 작업을 수행한다. MHEG 객체 관리기로 부터 필요한 MHEG 객체를 요청할 수 있으며 프리젠테이션 인터페이스로부터 사용자 사건을 입력 받고, 출력으로 동기 순서에 따른 MHEG런타임 내용 객체와 복합 프레임을 출력할 수 있다.

MHEG객체를 해석하기 위하여 프리젠테이션 인터페이스로부터 사용자 사건이 입력된다. 사용자 사건은 단순한 선택(예를 들어 버튼의 선택)이거나 입력을 동반한 사건(에디트 박스에서의 입력)일 수 있다. 메시지를 받은 링크/액션 처리기는 링크 조건등을 검사하며, MHEG런타임 객체 참조시 런타임 객체 처리기에게 런타임 객체를 요청한다. 런타임 객체 처리기는 입력된 런타임 객체를 풀에서 찾아 요청한 처리기에 전송한다. 만약 풀에 런타임 객체가 존재하지 않으면 MHEG객체 관리기에게 MHEG객체를 요청

한 후 런타임 객체를 생성한다. 생성된 런타임 객체를 풀에 삽입하고 런타임 객체를 요청한 처리기에게 전송한다. 해당 처리기는 나머지 작업을 수행함으로써 MHEG 객체를 해석한다.

나. 인터프리터를 위한 런타임 객체

런타임 객체는 MHEG객체가 가지고 있는 정보로부터 인터프리터 내부에서 사용되기 위해 "new" 액션을 사용하여 생성된다. 런타임 객체에는 런타임 스크립트(rt-script), 런타임 내용(rt-content), 런타임 멀티플렉스 내용(rt-multiplexed content), 런타임 복합(rt-composite) 객체가 있다. 런타임 스크립트 객체는 응용프로그램 수준에서 지원을 해주어야 하기 때문에 본 논문에서 제외한다. 또한 런타임 멀티플렉스 내용도 디멀티플렉스(demultiplex)과정이 필요하기 때문에 제외하였다.

본 논문에서 사용되는 런타임 객체는 런타임 복합 객체와 런타임 내용객체가 있다. 런타임 복합객체는 MHEG 복합 객체로 부터 생성되어지며 시공간 동기화 정보, 링크와 액션 정보 및 하위 구성요소로서 런타임 내용 객체를 포함하고 있다. 런타임 내용 객체는 응용 수준에서의 멀티미디어 데이터의 사용을 위하여 MHEG 내용 객체의 정보를 이용하여 생성되어진다. 즉, 하나의 런타임 내용 객체는 하나의 멀티미디어 데이터를 의미한다. 본 논문에서는 하나의 멀티미디어 화면을 구성하기 위하여 런타임 복합 객체를 사용하며, 멀티미디어 데이터를 위하여 런타임 내용 객체를 사용한다.

복합 객체는 기본적인 속성과 복합 행위에 대한 속성들을 가지고 있다. 또한 링크의 집합을 포함하고 런타임 내용 혹은 런타임 복합을 나타내는 요소들을 포함하며, 요소들의 시공간적 표현을 나타내기 위한 동기화 정보를 가지고 있다. 그러므로 런타임 복합 객체는 하나의 화면 단위를 의미한다.

런타임 내용객체는 MHEG 클래스의 내용 객체를 참조하여 설계하였으며, 응용프로그램에서 필요한 기능들, 즉, 에러 처리 및 장치 종속적인 요소들을 추가하여 MHEG 인터프리터내에서 사용할 수 있게 구성하였다. 클래스들은 모두 RT_OBJECT클래스로부터 상속을 받는다. 하위 클래스로는 TemporalObject, ProjectorObject, LinkObject가 있다. TemporalObject는 시간기반 객체들을 위한 상위 클래스이고, 시간적 요소들을 다루는 변수들을 포함하고 처리할 수 있는 함수들을 포함하고 있다. ProjectorObject는 공간상의 배치와 관계 있으며, 공간상의 배치를 요구하는 객체들을 위한 상위 클래스이고, 공간상의 표현에 필요한 속성들과 함수들을 가지고 있다.

LinkObject 클래스는 사용자와의 상호작용과 객체들간의 상호작용에 필요한 기능들을 가지고 있는 상위 클래스이다. 각각의 최하위 클래스들은 멀티미디어 데이터를 나타내며, 상위 클래스로부터 상속을 받게 되고, 자신의 미디어 특징을 표현하고 다룰 수 있는 속성과 기능들을 가지고 있다. 또한 런타임 내용 객체는 각자의 논리적 공간을 위한 채널을 가지고 있다.

런타임 내용 객체는 동기 처리 과정시 기본적인 메시지들과 에러에 대한 메시지를 동기 처리기에게 전달한다. 메시지는 {START, STOP, EVENT(Type), ERROR}의 4개로 구성되어 있다. START는 객체를 실행한다는 것을, STOP은 객체의 실행을 중지하고 화면에서 사라진다는 것을, EVENT(Type)는 이벤트의 종류를 매개변수로 하여 객체가 이벤트를 발생시켰다는 것을 동기 처리기에게 전달한다. ERROR 메시지는 객체 실행시 에러가 발생되었다는 것을 전달한다.

위에서 설명한 런타임 객체를 처리해 주는 런타임 객체 처리기는 MHEG객체로부터 런타임 객체의 생성과 관리를 수행하고 MHEG객체 관리기(handler)와 상호작용을 수행한다. 또한 수행중인 처리기가 런타임 객체 혹은 링크, 액션 객체에 대한 요청시 해당 객체를 처리기에 전달한다. 런타임 객체 처리기는 크게 2가지의 작업을 수행한다. 첫째 요청된 MHEG객체를 준비하는 일과 MHEG객체를 런타임 객체로 변환시켜주는 일이다. [알고리즘 2]는 전체적인 런타임 객체 처리 알고리즘이며, 런타임 객체 생성 처리는 MHEG 객체 관리기로부터 MHEG 객체를 받아 런타임 객체 변환후 풀에 추가한다.

[알고리즘 2] 런타임 객체 처리 알고리즘

```
RunTimeProcess()
{InputData();
    if(Kind(입력데이터)==런타임객체){//데이터종
    류판별
```

```
    SeekPool();// 런타임 풀을 조사
    if(풀에 객체가 있으면) Return 런타임 객체;
    else {// 런타임 객체가 없으면
      CallMHEGObjectHandler(MHEG_ID);
      Convert();//얻은 객체를 런타임 객체로 변환
      AddToPool();풀에 추가
      Return 런타임 객체;
    }
  }
 else {//입력 데이터의 종류가 링크/액션일 경우
    SeekPool();// 런타임 풀을 조사
    if(런타임 풀에 객체가 있으면) Return 객체;
    else {// 객체가 없으면
      CallMHEGObjectHandler(MHEG_ID);
      AddToPool();풀에 추가
      Return 객체;
    }
  }
}
```

런타임 풀은 생성된 런타임 객체를 저장하는 자료 구조로서 리스트와 배열의 조합 구조로 이루어져있다. 런타임 풀은 하나의 런타임 복합 객체를 단위로 구성되기 때문에 요소의 수가 검색 속도를 고려해야 할 만큼 크지가 않기 때문에 일반적인 리스트 구조를 이용한다.

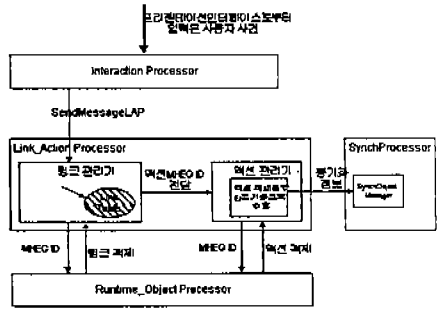
다. 상호작용 처리와 링크/액션

상호 작용에는 시스템에 대한 사용자의 단순한 명령으로 원하는 MHEG 객체의 실행을 수행하는 선택(예, 버튼 컨트롤)과 사용자의 정보 입력을 동반한 이벤트를 발생 시키는 수정(예, 텍스트 입력 박스)의 2가지 종류가 있다. 이러한 상호 작용 종류는 MHEG에서 제공하는 스타일을 가지고 있으며 버튼(button), 슬라이더(slider), 엔트리 필드(entry field), 메뉴(menu), 스크롤링 리스트(scrolling list) 등을 표준으로 지원한다. 이러한 표준 스타일 클래스에서 상속받은 새로운 스타일을 응용프로그램에서 정의하여 사용할 수도 있다.

상호 작용에는 시스템에 대한 사용자의 단순한 명령으로 원하는 MHEG 객체의 실행을 수행하는 선택(예, 버튼 컨트롤)과 사용자의 정보 입력을 동반한 이벤트를 발생시키는 수정(예, 텍스트 입력박스)의 2가지 종류가 있다. 이러한 상호 작용 종류는 MHEG에서 제공하는 스타일을 가지고 있으며 버튼(button), 슬라이더(slider), 엔트리 필드(entry field), 메뉴(menu), 스크롤링 리스트(scrolling list) 등을 표준으로 지원한다. 이러한 표준 스타일 클래스에서 상속받은 새로운 스타일을 응용프로그램에서 정의하여 사용할 수도 있다.

상호 작용 처리기가 발생하는 메시지는 2가지이며, 선택을 위한 "SELECTION", 수정을 위한 "MODIFICATION"이 있다. 각각의 메시지들은 행동 상태와 행동을 변화시키는 액션, 그리고 행동을 조사할 수 있는 액션의 3가지의 부류로 이루어져 있다. 사용자의 상호 작용의 결과는 새로운 링크 객체를 실행하거나, 새로운 액션 객체를 호출하는 작업을 요구한다. 따라서 상호 작용 처리는 링크/액션 처리와 밀접한 관계를 가지고 있다.

링크/액션 처리기는 MHEG 링크 객체와 액션 객체를 처리해준다. 2개의 관리기로 나뉘어지며, 액션 관리기와 링크 관리기가 있다. 이 두개의 관리기는 서로 밀접한 관계를 유지하며 상호 작용을 한다. MHEG에서의 링크는 하이퍼텍스트에서의 링크와는 다르게 사건기반 트리거(trigger)와 비슷한 의미로 사용된다[Buf94]. 이러한 MHEG 링크 객체는 상호 작용을 지원하기 위한 방법을 제공한다. 이전의 MHEG 객체의 상태가 변화되었을때 하나 혹은 그 이상의 링크들이 개시(fiable)될 수 있으며, 결과적으로 링크/액



(그림 8) 상호작용 처리기와 링크/액션 처리기와의 데이터 흐름도

(Fig. 8) Data flow between interaction processor and link/action processor

션 처리기의 가장 중요한 기능이 이러한 링크 개시를 감지하고, 뒤따르는 액션을 적절히 실행하는것이다. MHEG표준에는 많은 양의 링크 객체들이 있기때문에 상태가 변화되었을때 링크의 개시와 관련된 효율적인 수단들이 있다. (그림 8)은 상호작용 처리기와 링크/액션 처리기와의 데이터 흐름도이다.

링크 관리기의 역할은 링크 객체를 관리하고 처리하는 것이다. 링크 객체가 준비 되어지고, 모든 소스와 타겟 참조들이 링크 관리기에 의해 접근 가능하면, 액션들이 여러 타겟에 수행되어질 수 있다. 링크 관리기는 링크 상태 테이블에 객체정보와 평가 상태를 유지하고 있다. 준비되어진 모든 객체는 테이블안에 첨가 되어질 수 있으며, 객체가 파괴되어질 때까지 테이블 안에 있을 수 있다. 일단 개시 조건이 만족되면, 액션은 링크 객체의 타겟에 대한 액션 관리기를 호출함으로써 타겟에 대한 액션을 수행한다.

링크 알고리즘은 MHEG 런타임 객체의 조건을 검사하는 과정과 조건이 만족되었을 경우 해당 액션을 처리해주는 부분으로 나눌 수 있다.

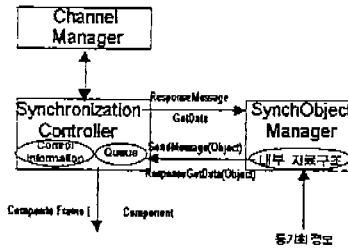
첫번째 수행 과정은 런타임 풀로 부터 링크 객체를 얻은 후 링크조건을 테스트한다. 테스트 결과가 참이면 해당 액션 객체의 ID를 액션 관리기로 전송한다. 만약 테스트 결과가 거짓이면 아무일도 하지 않는다.

액션 관리기는 입력된 액션 ID를 이용하여 런타임 객체 풀에서 전달 받은 액션 객체를 이용하여 동기화 정보를 추출하는 것이 가장 큰 기능이다. 동기화 정보 추출은 MHEG 액션 객체로 부터 기정의된 동기화 정보 구조로의 변환을 의미한다. MHEG 액션 객체 타겟은 주로 MHEG 런타임 복합 객체이지만 단일 런타임 내용 객체일수도 있다.

라. 동기 처리기

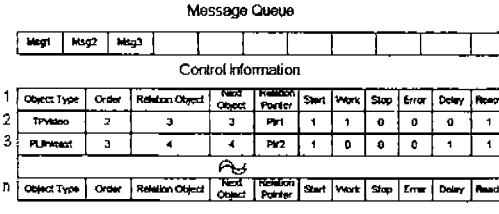
멀티미디어 동기화를 위한 통합 계층 프레임워크에 관한 연구[16]가 진행되고 있으며. MHEG 내에서의 동기화는 인터 스트림 동기화(inter-stream synchronization)인 13가지의 시간 동기화 방법을 정의하고 있는데, 이 방법은 [17]의 연구와 같다. 동기 처리기는 런타임 컴포넌트를 사용자에게 표현해 주며, 사용자와의 인터페이스를 위하여 시스템상의 표현 서비스와 상호 작용을 수행한다. 동기 처리는 기본적인 MHEG 표현 액션을 처리하며, 이러한 액션은 동적 채널 할당, 지각(perceptability), 시간, 공간, 음향, 런타임 멀티플렉스상의 스트림 선택과 같은 런타임 컴포넌트의 해석(rendition)에 영향을 미친다. 또한 동기처리는 동기화 정보를 이용하여 객체들을 표현할 수 있어야 한다.

기본적으로 동기 처리기는 동기화 정보내에 포함되어 있는 MHEG 런타임 객체간의 시공간적 정보와 링크, 액션 정보등을 이용하여 기능을 수행한다. 또한 사용자에게 객체 효과를 나타내기 위하여 프리젠테이션 인터페이스와 상호 작용을 한다. (그림 9)와 같이 동기 처리기는 3개의 관리기, 즉 동기 제어기, 동기 객체 관리기, 채널 관리기로 구성된다.



(그림 9) 인터프리터를 위한 동기 처리기의 구성도
(Fig. 9) Configuration of synchronization processor for interpreter

동기 제어기는 크게 두개의 자료 구조와 하나의 메시지 처리 알고리즘으로 이루어졌으며, 서브시스템의 메시지를 담고 있는 메시지 큐와 START메시지를 발생한 객체의 정보를 가지고 있는 제어 정보로 구성된다. (그림 10)은 동기처리기의 내부 자료 구조이다.



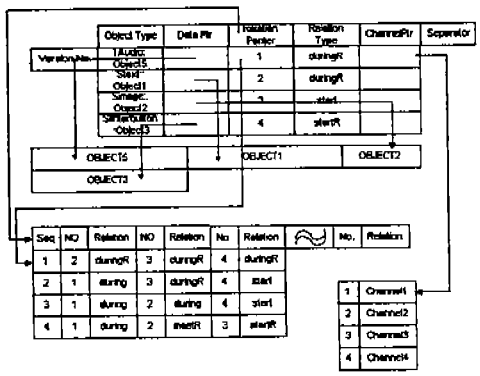
(그림 10) 동기 처리기의 내부 자료구조
(Fig. 10) Internal data structure of synchronization processor

메시지 큐(Message Queue)는 메시지 처리기에서 전달되는 메시지들을 저장하는 큐이다. 메시지의 형

식은 MessageType(Parameter)으로 구성되어 있다. 런타임 내용 객체는 실행시 메시지를 동기 제어기에게 전달하여 처리할 수 있게 해 준다. 제어 정보는 Start 메시지를 발생한 객체들의 정보들을 관리하는 자료구조로써 객체들의 상태는 {Start, Work, Stop, Error, Delay, Ready}로 구성되어 있다. 동기 제어기는 런타임 객체의 상태를 제어 정보를 이용하여 검색할 수 있고, 각각의 상태와 객체간의 링크 정보를 참조하여 다음의 행위를 결정한다.

런타임 복합 객체는 내부적으로 포함한 내용 클래스의 시공간적 정보, 액션정보와 링크정보를 포함하고 있다. 따라서 액션 객체에 의해 추출된 동기정보를 동기 처리기에서 이용한다. 하나의 런타임 내용 객체는 시간상에 존재하는 모든 객체와의 시간 관계성을 하나의 데이터 구조로 유지하고 있다. 시간 관련성 테이블은 동기 처리기에 의해 참조되고, 객체간의 시간 동기화를 유지하기 위해 사용된다. 공간 관련성 테이블은 내부 객체와 관련이 있는 채널들을 포함한다. (그림 11)은 미디어들간의 동기 정보를 위한 데이터 구조를 나타낸다. 런타임 복합 객체의 경우 객체간의 시간적 관계는 RelationType이라는 변수형으로 나타내고, MHEG의 기본적 동기 관련성을 나타낸다. 여기서 R접미어는 역관계를 의미한다.

RelationType{ before, meets, overlaps, during, start, finishes, equals, beforeR, meetsR, overlapsR, duringR, startR, finishesR};



(그림 11) 미디어들간의 동기 정보를 위한 데이터 구조
(Fig. 11) Data structure for synchronization information between media

동기 처리 알고리즘은 동기 제어기의 내부 자료 구조를 이용하여 런타임 내용 객체의 동기를 맞추어 사용자에게 보여 주는 루틴이며, 동기 메시지 처리 알고리즘은 런타임 객체간의 관계 정보를 이용하여 멀티미디어 객체의 정확한 시간 관계성을 이용하여 화면에 보여주는 부분이다. 입력된 메시지 종류에 따라 각각의 메시지 처리를 수행하며, QUIT 메시지 입력시 동기 메시지 처리를 종료한다. 객체간의 시간적 관계 처리는 RelationCheck()함수에서 처리해주며, 메시지 처리전에 동기 관리기로 부터 지연 신호를 입력받은 객체를 처리해 준다. 동기 메시지 처리 알고리즘은 [알고리즘 3]과 같다.

[알고리즘 3] 동기 메시지 처리 알고리즘

```
Start {
while(Message != QUIT)
{
  if(In Control Information, Delay==TRUE)
    ProcessDelay();//지연 객체가 있으면 처리
    MessageInput()//메시지입력이 있을때까지 대기
    switch(Message Type) {
      START:
        IsValid( START Message)
        ControlInformationUpdate();
        if(GetLink()==NULL) continue;
        if(RelationCheck()==OK)
         Return START MESSAGE
            to SynchObject Manager
        else {
         if(RelationCheck()==DELAY)
              Return DELAY MESSAGE
                to SynchObject Manager
         else
              Return ERROR MESSAGE
                to Error Processor Interface
      }
    STOP:
      IsValid(STOP MESSAGE)
      ControlInformationUpdate();
      if(RelationCheck()==OK)
        Return STOP MESSAGE
```

```
            to SynchObject Manager
    else {
      if(RelationCheck()==DELAY)
            Return DELAY MESSAGE
                to SynchObject Manager
      else
        Return ERROR MESSAGE
            to Error Processor Interface
    }
  EVENT:
    IsValid(EVENT MESSAGE)
    ControlInformationUpdate();
    if(GetLink()==NULL) continue;
    if(GetAction()==NULL) continue;
    ProcessEvent()
    Return ResponseEvent MESSAGE
        to SynchObject Manager
  ERROR:
    IsValid(ERROR MESSAGE)
    Return ERROR MESSAGE
        to Error Processor Interface
    DEFAULT:
        continue;
    } // End of switchs
  } // End of while
} END
```

채널 관리기는 채널들의 정보를 이용하여 해당 기능을 수행한다. 채널은 런타임 객체의 속성을 이용하여 객체를 해당 물리적 장치로 사상하는 논리적 공간이다. 물리적 장치는 원격 컴퓨터, 화면, 스피커 등이 될 수가 있다. 채널 관리기는 공간상의 표현을 필요로 하는 런타임 객체의 크기를 재조정할 수 있으며, 오디오 객체는 음향의 정보를 이용하여 해당 물리적 장치에 적절하게 출력될 수 있다.

4. 결 론

본 논문에서는 멀티미디어/하이퍼미디어 응용을 위한 MHEG 엔진을 설계하였다. 설계한 MHEG 엔진은 MHEG 객체의 생성과 프리젠테이션 기능을 수행하는 네개의 모듈로 구성된다. 설계한 MHEG 엔진에서, 전송된 MHEG 객체에 대한 디코딩후의 내부 포멧을 각각의 MHEG 객체가 가지는 객체 지향 방식의 표현 구조와 사상되는 구조로 정의함으로써, 디코딩 및 인터프리터의 MHEG 객체에 대한 해석의 용이성을 가져왔으며, MHEG 객체 관리자 모듈에서는 이러한 객체 정보를 계승과 포함 관계를 쉽게 표현할 수 있는 트리 형태로 관리함으로써, 외부 객체나 데이터 화일에 대한 동적인 참조 및 서브 객체에 대한 관리가 용이하도록 설계하였다. 또한 MHEG 표준에서 제공하는 동기화 정보를 이용하여 멀티미디어 객체간의 공간적 표현을 채널을 이용하여 처리하였고 시간적 관계성을 Allen이 제안한 13가지 시간관계성에 따라 객체간의 시간 관계성을 묶었으며, 또한 동기화 정보를 포함하는 복합 객체를 처리할 수 있는 객체 지향 동기화 모듈과 동기 메시지 처리 알고리즘을 제안하였다.

MHEG 표준이 가공하기 어려운 멀티미디어 데이터에 대한 공유와 실시간 전송을 해결하기 위한 새로운 제안이라는 측면을 고려해 볼 때, 제안한 MHEG 엔진은 차후 멀티미디어 응용 분야에서 기반이 될 수 있다는 점에서 그 의미를 둘 수 있겠다. 설계된 엔진은 Windows NT 환경에서 Visual C++로 구현중이며, 초고속 통신 응용 기술 프로젝트에 사용될 예정이다[18]. 추후 연구 내용은 MHEG 엔진을 기반으로 하는 응용을 쉽게 개발하기 위한 응용 프로그래밍 인터페이스 및 데이터베이스 인터페이스를 포함한 개발 환경으로의 확장이다.

참 고 문 헌

[1] R. Steinmetz and K. Nahrstedt, Multimedia: Computing, Communications & Applications, Prentice Hall Inc., 1995.

[2] M. B. Thomas and E. Wolfgang, MHEG: Explained, IEEE Multimedia, Spring, 1995, pp. 26-38.

[3] J. F. K. Buford, Multimedia Systems, ACM Press, 1994.

[4] B. Furht and M. Milenkovic, A Guided Tour of Multimedia Systems and Applications, IEEE

Computer Society Press, pp. 147-284, 1995.

[5] ISO/IEC DIS 13522-1 Information technology-Coding of Multimedia and Hypermedia information-Part 1 : MHEG object representation-Base notation(ASN. 1), 1994. 10.

[6] R. Price, MHEG : An Introduction to the Future International Standard for Hypermedia Interchange, Proc. of the 1st ACM International Conference on Multimedia, Anaheim(CA), USA, Aug. 1-6 1993, pp. 121-128.

[7] F. Colaitis and F. Bertrand, "The MHEG Standard : Principles and Examples of Applications," Proceedings of the Eurographics Symposium, (eds.) W. Herzner and F. Kappe, Springer-Verlag, pp. 3-17, 1994.

[8] J. F. K. Buford and C. B. Gopal, "Standardizing a Multimedia Interchange Format : A Comparison of OMFI and MHEG," Proceedings of the International Conference on Multimedia Computing and Systems, IEEE Computer Society Press, pp. 463-472, 1994.

[9] Tomas Meyer-Boudnik, Wolfgang Effelsberg MHEG : An Interchange Format for interactive Multimedia Presentations , IEEE Multimedia Magazine 1995.

[10] J. J. Sung, M. Y. Huh, H. J. Kim, and J. H. Hahm, Hypermedia Information Retrieval System Using MHEG Coded Representation in a Networked Environment, Proc. of the 2nd International Workshop on Multimedia : Advanced Teleservices and High-Speed Communication Architectures, Heidelberg, Sep. 26-28, 1994, R. Steinmetz (Ed.), Springer LNCS vol. 868, 1994, pp. 67-77.

[11] C. Bertin, "Eurescom IMS1 Projects(Integrated Multimedia Services at about 1 Mbit/s)," Proceedings of the Second International Workshop on Multimedia : Advanced Teleservices and High-Speed Communication Architectures, IWACA '94, (Ed.) R. Steinmetz, Springer-Verlay, pp. 53-66, 1994.

[12] F. Kretz and F. Colaitis, "Standardizing Tour of Multimedia Systems and Applications, (Reprinted from IEEE Communications Magazine, pp. 60-70, 1992), (Eds.) B. Furht and M. Milenkovic, IEEE Computer Society Press, pp. 402-412, 1995.

[13] A. Rizk, F. Malezieux and A. Leger, "Distributed Hypermedia Link Service on WAN : An Environment with MHEG on the ATM Network," Proceedings of the Eurographics Symposium, (eds.) W. Herzner and F. Kappe, Springer-Verlag, pp. 18-28, 1994.

[14] ISO/IEC IS 8824 Specification of Abstract Syntax Notation One (ASN. 1). Second edition. 1990.

[15] ISO/IEC IS 8825 Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN. 1). Second edition. 1990.

[16] Nikos B. Pronios, Theodoros Bozios, "Multimedia and Hypermedia Synchronization : A Unified Framework," Proceedings of the Second International Workshop on Multimedia : Advanced Teleservices and High-Speed Communication Architectures, IWACA'94, (Ed.) R. Steinmetz, Springer-Verlag, pp. 153-166, 1994.

[17] J. F. Allen, Maintaining Knowledge About Temporal Intervals , Communications of the ACM, Vol. 26, No. 11, pp. 832-843, 1983. 11.

[18] 이세훈, 왕창종외, "초고속 정보 통신망에서 원격 교육을 위한 개방형 시스템," 한국정보과학회 가을학술논문발표집, Vol. 22, No. 2, pp. 765-768, 1995.

이 세 훈

1985년 인하대학교 전자계산학과 졸업(이학사)

1987년 인하대학교 전자계산학과 대학원(이학석사)

1996년 인하대학교 전자계산학과 대학원(공학박사)

1987년~1990년 해병대 전산실 분석장교

1993년~현재 인하공업전문대학 전자계산기과 조교수

관심분야: 멀티미디어/하이퍼미디어, 개방형 시스템, 소프트웨어 공학, 교육용 소프트웨어

왕 창 종

1964년 고려대학교 물리학과 (학사)

1975년 성균관대학교 (경영학석사)

1981년~1990년 인하대학교 전자계산소장

1992년~1993년 정보과학회 부회장, 전산교육연구회 위원장

1979년~현재 인하대학교 전자계산공학과 교수

관심분야: Software Engineering, CASE, 역공학, Expert System, Intelligent Tutoring System