J. of Basic Sciences, Cheju Nat. Univ.
(2), 35~50, 1996

기초과학연구 제주대학교
9(2), 35~50, 1996

우선 순위를 이용한 멀티미디어 프리젠테이션

(A Multimedia Presentation using Priority)

김한일 · 김성백
제주대학교 컴퓨터교육과

이봉규
제주대학교 전산통계학과

요 약

본 논문에서는 멀티미디어 프리젠테이션 시스템에서 서로 다른 미디어간에 발생하는 동기의 어긋남을 방지하기 위해 객체의 손실을 허용하는 멀티미디어 프리젠테이션 시스템을 제안하며, 이를 구성하는 시나리오 기술 언어(SDL, Small scenario Description Language)와 동기화 기법을 설계, 구현하였다. 제안된 시스템은 우선 순위가 낮은 객체를 희생시킴으로써, 어긋남이 발생하지 않고 한정된 시스템 자원내에서 수행가능한 프리젠테이션을 제공한다.

1. 서 론

최근들어 컴퓨터와 사용자 간의 인터페이스가 강조되면서 멀티미디어 데이타를 처리하기 위한 많은 연구가 진행되고 있다. 멀티미디어 프리젠테이션 시스템은 디스크에 독립적으로 저장된 모노미디어 데이타를 사용자가 기술한 시나리오에 따라 프리젠테이션 시에 동기화 시키는 시스템으로 정의될 수 있다. 따라서 사용자는 마치 하나의 멀티미디어 화일이 처리되는 것처럼 느끼게 된다[3][11].

이러한 멀티미디어 프리젠테이션 시스템의 가장 큰 연구 과제는 동기화 기법이다. 기존에는 주로 미디어 객체별로 출력 프로세스를 하나씩 생성함으로써 동기화를 수행하였으나, 동기화에 필요한 계산 시간과 데이타 전송 시간으로 인해 동기가 어긋나는 문제점이 있다[2][3][5]. 이러한 문제점을 해결하기 위해[5]에서는 프리젠테이션 이전에 데이타를 버퍼로 읽어들이는 기법을 도입했으나, 버퍼의 크기에도 제한이 있으므로 이 방법 역시 한계가 있다. [2]에서는 동기화를 위해 모노미디어 데이타의 일부를 누락 혹은 중복시키는 기법을 이용하였으나, 이 방법은 예상치 못한 데이타의 손실을 가져올 수 있다[3].

본 논문에서는 프리젠테이션 시에 발생하는 미디어간의 어긋남을 방지하기 위해 객체의 손실을 허용하는 멀티미디어 프리젠테이션 시스템을 제안한다. 객체의 손실이란 어긋남의 발생 원인인 시스템 부하를 줄이기 위해, 멀티미디어 객체에 부여된 우선 순위에 따라 객

체의 전부, 혹은 일부를 프리젠테이션 대상에서 제외시키는 것을 의미한다. 객체 손실을 허용하는 멀티미디어 프리젠테이션 시스템은 그림 1과 같은 구조를 지니며, 본 논문에서는 이중 시나리오 기술 언어, SDL과 동기화 기법을 설계, 구현하였다.

SDL은 스크립트 형태를 지니며, 객체 간의 기본적인 시간 관계와 우선 순위 표현 기능을 제공한다. SDL을 이용하여 기술된 사용자 시나리오는 동기화 과정에 객체간의 시간 관계와 우선 순위를 효과적으로 전달하기 위해 내부적으로 우선 순위 구간을 이용하여 저장된다.

동기화 기법은 프리젠테이션 이전에 모든 동기화 과정을 끝마치는 오프라인(off-line) 방식의 스케쥴러에 의해 수행된다. 스케쥴러는 우선 순위 구간별로 객체의 분할, 초기 배치, 그리고 전이(migration) 과정을 차례로 수행하며, 자원이 부족한 경우에는 우선 순위 구간에 존재하는 객체들의 일부를 프리젠테이션 대상에서 제외시킨다.

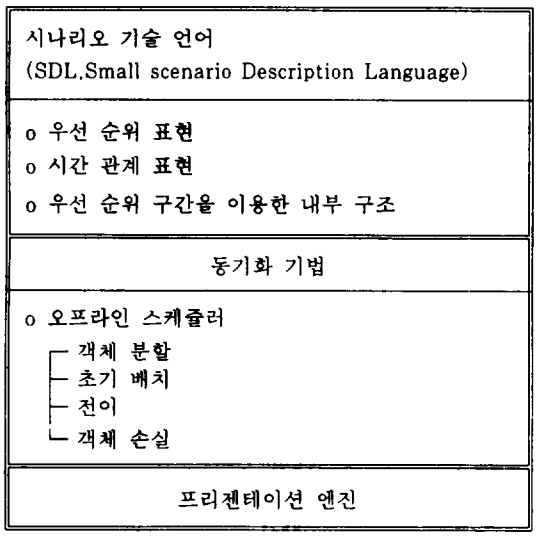

그림 1. 객체 손실을 허용하는 멀티미디어 프리젠테이션 시스템의 구성

2. 관련 연구

2.1. 시나리오의 기술

시나리오 기술이란 멀티미디어 객체와, 이들 간의 시간 관계를 표현하는 것을 의미한다. 현존하는 시나리오 기술 방식은 크게 테이블 형태, 그래픽을 이용한 형태 그리고 텍스트 형태가 있다.

○ 테이블 형태의 시나리오 기술

테이블 형태의 언어는 멀티미디어 객체와 객체의 속성, 예를 들어 데이타 화일의 이름, 프리젠테이션 시작 및 종료 시간, 그리고 특수 효과 등을 테이블로 구성한 것이다. 이러한 언어의 대표적인 예는 IBM에서 개발한 REXX(Systems Application Architecture™ 용 프로그래밍 언어)의 변형 형태인 AVA 언어이다[7]. 테이블 형태의 언어는 각 객체별로 프리젠테이션되는 형식을 쉽게 파악할 수 있는 장점이 있으나, 전체적인 시나리오 구조를 이해하기가 어려워 현재 많이 사용되지는 않는다.

○ 그래픽을 이용한 시나리오 기술 언어

그래픽 이용한 언어는 미디어 객체들 사이의 시간 관계를 시간 축을 이용하여 그래픽하게 표현한 형태를 지닌다. 이러한 방법은 구현하기가 어렵고 세밀한 부분을 조정하는데는 불편한점이 있으나, 사용자가 전체 시나리오 구조를 한 눈에 알아볼 수 있기 때문에 많이 이용되고 있다[5][8]. 현재 상용화된 시스템으로는 MACROMIND사의 ACTION!이 있다.

○ 텍스트 형태의 시나리오 기술 언어

텍스트 형태의 언어는 말 그대로 멀티미디어 객체와 이들 사이의 시간 관계를 프로그래밍 언어 형식의 텍스트 화일로 기술한 형태를 지니는데, 보통 이를 스크립트라 한다. 스크립트중에는 실제 프로그래밍 언어의 형식을

빌린 것도 있으며, 사용자가 이해하기 쉽도록 자연어 형태를 지닌 것도 있다. [10]에서는 concurrent programming에서 많이 이용되는 path expression을 이용하였으며, [4]에서는 ADA형식으로 시나리오를 기술하였다. [1]은 애니매이션을 위한 논리 언어 형태의 스크립트 언어를 새로 정의 하였다. 스크립트는 의미가 명확하고, 표현이 간결하며, 다루기가 편리하고, 또한 구현하기가 쉽기 때문에 현재 가장 많이 이용되고 있다.

2.2. 동기화 기법

○ 하드웨어적 동기화 기법

멀티미디어 PC에서 널리 사용되고 있는 CD-ROM 타이틀은 데이타를 물리적으로 배치하는 동기화 기법을 이용하고 있다. 물리적 데이타 인터리빙이라 불리는 이 기법은 프리젠테이션 시 별도의 동기화 작업이 필요하지 않으므로 좋은 결과를 얻을 수 있으나, 시나리오를 사용하지 않기때문에 그 내용을 변경할 수 없는 단점이 있다.

○ 프리젠테이션 프로세스를 이용한 기법

물리적 데이타 인터리빙대신 논리적인 데이타 인터리빙을 쓰는 시스템도 있다. 논리적 데이타 인터리빙이란, 객체의 데이타는 디스크에 독립적으로 위치하며, 프리젠테이션 시에 이들을 읽어들여 동기화 작업을 수행하는 것으로, 사용자에게는 마치 물리적 인터리빙에 의한 프리젠테이션인 것 처럼 느끼게 한다. [2]에서는 프리젠테이션을 담당하는 프로세스를 생성하여 논리적 데이타 인터리빙을 처리한다. 실제로 이 프로세스는 내부적으로 객체마다 별도의 프로세스를 생성하는데, 결국 이들의 생성 시점과 종료 시점이 동기화 시점이 된다. 그러나 프로세스의 생성과 종료 시간에는 이를 담당하는 운영 체제의 수행 시간과, 데이타를 원하는 출력 장치로 전달하는 시간이 포함되기 때문에 실제 프리젠테이션 시에는 미디어간에 어긋남이 발생하게 된다 [2][3][5].

○ 선반입(prefetching) 기법

[5]는 분산 환경에서의 멀티미디어 시스템을 구성하고, 네트워크 상의 지연 시간을 처리하기 위해 선반입 기법을 도입하였다. 선반입된 데이타는 버퍼링되므로 많은 시스템 메모리를 필요로 하며, 그 크기는 시스템의 성능에 따라 가변적이다.

○ 분할 스케쥴링 기법

멀티미디어 시스템에서는 동시에 여러 객체의 데이타를 프리젠테이션해야 하는데 반해, 일반적인 컴퓨터 시스템은 한순간에 하나의 작업만을 수행한다. 이를 위해 여러 객체를 번갈아 출력하여, 실제로는 순차적인 작업을 하지만 사용자에게는 동시에 수행하는 것처럼 보이도록 한다. [12]에서는 시스템 자원을 공평하게 분배하기 위해 각 객체를 몇 개의 구간으로 나누어 스케쥴링 한다.

3. SDL의 설계

3.1. 요구 조건

일반적으로 시나리오 기술 언어는 객체 간의 시간 관계와 사용자와의 동적 인터페이스, 반복 수행, 그리고 조건부 수행 등의 다양한 기능을 제공해야 하나, SDL은 동기화 과정에 필요한 우선 순위의 기술에 목적을 두었기 때문에 객체 간의 시간 관계와 우선 순위 표현 기능만을 제공한다.

가. 시간 관계

두 개의 시간 간격 사이에는 그림 2와 같이 모두 13 가지의 시간 관계가 존재한다[6]. 그림 2에서 equals를 제외한 나머지 6 가지의 시간 관계에는 역관계가 존재하기 때문에,

총 13 가지가 된다. 시나리오에 기술된 모든 객체의 시간 관계는 이들의 조합에 의해 표현 가능하므로, SDL은 기본적으로 이 13 가지 시간 관계를 표현할 수 있어야 한다.

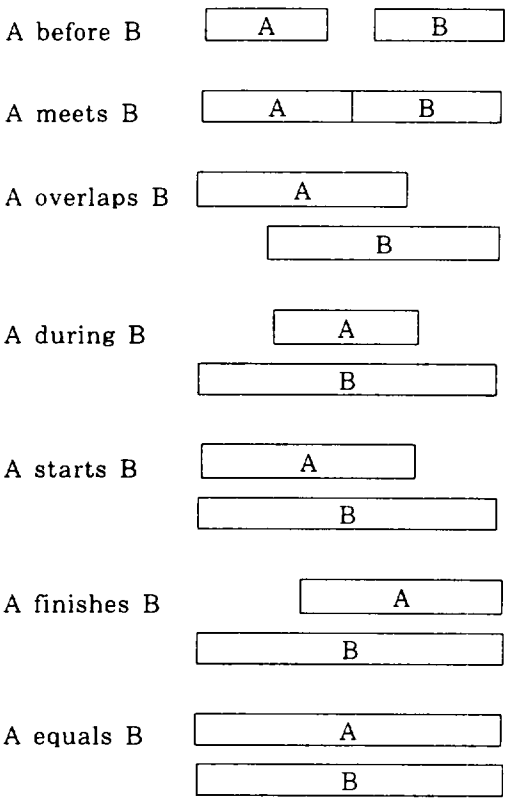

그림 2. 두 개의 시간 간격에 대한 13 가지 시간 관계

나. 우선 순위

시나리오에 기술된 객체들은 자원, 특히 메모리가 부족할 경우 객체간의 경쟁에 의해 프리젠테이션 과정에서 동기의 어긋남이 발생할 수 있다. 이러한 문제을 해결하기 위해 기존 연구에서는 일부 데이타에 대한 누락(drop) 또는 중복(duplicate) 기법을 도입하였으나, 이와 같은 작업은 의도와는 달리 사용자가 중요하게 생각하는 데이타의 예상치 못한 손상을 유발시킬 수 있다[2][3].

이와 같은 예상치 못한 데이타의 손실은 누락 또는 중복 작업이 사용자의 동의 없이 이루어지기 때문에 발생한다. 이 문제는 SDL에 우선 순위 표현 기능을 추가하여 사용자가 객체의 중요성을 표현하게함으로써 해결될 수 있다. 객체에 우선 순위를 부여함으로써 다음과 같은 잇점을 얻을 수 있다.

○ 희생자 선택

스케쥴러는 충돌이 발생한 시나리오 객체 가운데 우선 순위가 낮은 객체를 희생자로 선정한다. 희생자로 선정된 객체는 우선 순위에 따라 스케쥴링 대상에서 완전히 제외되거나, 충돌이 발생하지 않는 범위 내에서 일부가 수행될 수 있다. 이와 같이 우선 순위를 이용하여 희생자를 선택함으로써, 기존에 발생했던 예기치 못한 데이타의 손실을 막을 수 있다.

○ 확장된 admission control

시스템 자원이 추가의 프리젠테이션 작업을 수행할 만큼 충분하지 않을 경우, 기존의 addmission control 알고리즘은 이 작업을 허용하지 않는다. 그러나 사용자가 원할 경우 낮은 우선 순위를 가지고 있는 객체의 일부를 희생함으로써 비록 완전하지는 않지만 새로운 프리젠테이션을 추가로 수행할 수 있다.

○ 손상된 데이타의 처리

특정 객체의 데이타가 손상되어 정상적인 처리를 할 수 없는 경우에 기존에는 시나리오에서 해당 객체에 대한 부분을 완전히 고쳐야 했으나, 객체에 우선 순위가 부여된 경우에는 단지 우선 순위를 낮춤으로써 전체 시나리오의 수정 없이도 손상된 객체를 프리젠테이션 대상에서 제외 시킬 수 있다. 또한 손상된 객체가 복구되었을 경우에도 우선 순위를 상향 조정함으로써 쉽게 프리젠테이션 대상에 포함

시킬 수 있다. 이는 특정 우선 순위를 손상된 데이타의 처리를 위해 설정함으로써 구현할 수 있다.

○ 시나리오의 수행 가능성 예측

우선 순위의 도입과 오프라인 스케쥴링으로 프리젠테이션에 필요한 자원의 양을 미리 계산 할 수 있다. 따라서 사용자는 작성한 시나리오가 수행 가능한지를 사전에 판단하여, 주어진 시스템 환경에 적합한 시나리오를 생성할 수 있다.

3.2. SDL의 구문

시나리오를 구성하는 객체의 특성에는 정적 특성과 동적 특성이 존재한다. 정적 특성은 객체가 생성될때 부여되는 특성으로 일반적으로 객체가 저장된 화일의 헤더 부분에 기록되며 추후 변경이 불가능하다. 반면 동적 특성은 객체가 시나리오에 정의될 때 부여되는 특성으로, 사용되는 시나리오의 목적에 따라 얼마든지 변경이 가능하다. 그림 3은 객체의 정적 특성과 동적 특성을 보여준다.

미디어 종류	정적 특성	동적 특성
오디오	sampling rate, sample size	우선 순위 프리젠테이션 기간
비디오	프레임 개수/s, 프레임 크기	
이미지	압축률, 이미지 크기	

그림 3. 데이타 화일의 헤더에 기록된 모노미디어 객체의 정적 특성

SDL은 이러한 정보 중 동적 특성과 객체간의 시간 관계를 표현하기 위해 사용된다. SDL로 작성된 시나리오는 그림 4와 같이 크게 객체 정의 부분과 시간 관계 정의 부분으로 구성되어 있다.

```
#OBJECT
        시나리오 객체 정의 부분
#END
#RELATION
        시간 관계 정의 부분
#END
```

그림 4. SDL로 작성된 시나리오의 구조

3.2.1. 객체 정의 부분

시나리오 객체 정의 부분은 시나리오 객체의 동적 특성을 기술하는 부분이다. 시나리오 객체의 동적 특성에는 데이타가 저장된 화일 이름과, 프리젠테이션 기간, 그리고 우선 순위가 포함된다. 시나리오 객체의 동적 특성은 그림 5와 같은 형식으로 기술된다.

name: F(*file*) M(*mtype*) [P(*priority*)]

name : 객체의 이름
file : 객체의 데이타가 저장된 화일의 이름
mtype : 미디어 종류
priority : 객체의 우선 순위

그림 5. 시나리오 객체의 동적 특성 기술 구문

3.2.2. 시간 관계 정의 부분

3.1. 절에서 언급한 13 가지 시간 관계는 객체와 객체 사이의 빈 구간을 나타내는 시간 객체를 이용하여 보다 간단하게 표현할 수 있다[5][10]. 그림 6은 13 가지 시간 관계를 시간 객체와 equals, meets 시간 관계를 이용하여 표현한 것이다. 그림 6에서 T(δ)는 길이가 δ인 시간 객체를 나타낸다.

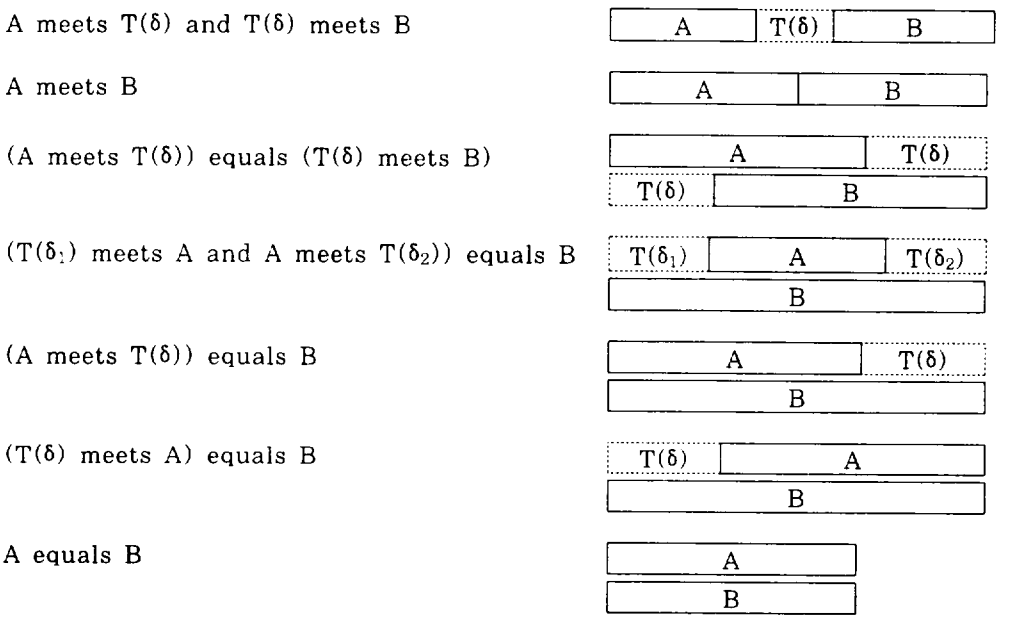

그림 6. 시간 객체, meets, equals를 이용하여 표현한 13 가지 시간 관계

Sequential(O_1, O_2, O_3, ..., O_n) ($2 \le n$)

$\Leftrightarrow$ O_1 Meets O_2 and O_2 Meets O_3 and ... and O_{n-1} Meets O_n

Parallel(O_1, O_2, O_3, ..., O_n) ($2 \le n$)

$\Leftrightarrow$ $\forall ij$ (O_i Equals O_j) $\quad 1 \le i \le n,\ 1 \le j \le n,\ i < j$

O_i :
- 시간 객체 — T(*duration*)
- 미디어 객체 — *object-name*(*duration*)
- 복합 객체
  - Sequential(O_1, O_2, O_3, ..., O_m) ($2 \le m$)
  - Parallel(O_1, O_2, O_3, ..., O_k) ($2 \le k$)

그림 7. n-ary 시간 관계를 표현하는 Sequential과 Parallel의 정의

그런데, equals와 meets는 이진 시간 관계(binary time relation)이기 때문에, 이를 이용하여 시나리오를 작성하면 시나리오가 길어지고 복잡해지는 단점이 있다. 본 논문에서는 이러한 점을 고려하여 그림 7과 같이 임의의 n개 사이의 시간 관계를 표현할 수 있는 Sequential과 Parallel를 정의하였다.

Sequential과 Parallel은 객체 간의 시간 관계를 나타내지만, 그 자체가 또다른 하나의 객체를 나타내는데, 이를 복합 객체(complex

object)라 한다. 복합 객체를 제외한 모든 객체는 프리젠테이션 기간을 괄호 속에 표시하며, 복합 객체의 프리젠테이션 기간은 계산에 의해 쉽게 얻을 수 있으므로 별도로 표시하지는 않는다. 시나리오 객체 A의 프리젠테이션 기간을 D(A)라 할때, 복합 객체의 프리젠테이션 기간은 다음과 같이 정의된다.

$$D(\text{Sequential}(O_1, O_2, \ldots, O_n)) = D(O_1) + D(O_2) + \ldots + D(O_n) \quad (2 \le n)$$

$$D(\text{Parallel}(O_1, O_2, \ldots, O_n)) = D(O_1) = D(O_2) = \ldots = D(O_n) \quad (2 \le n)$$

복합 객체를 이용하면 기존에 작성된 시나리오를 다른 시나리오에 포함시키거나 복잡한 객체를 단순화 시키는 등의 장점이 있다. 그림 8은 그림 6을 본 논문에서 정의한 Sequential과 Parallel을 이용하여 표현한 것이다.

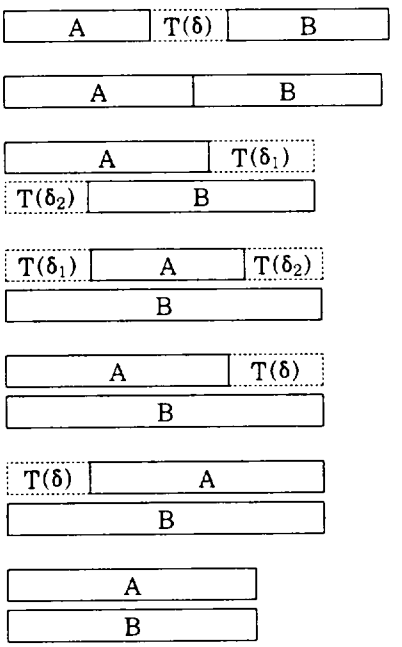

그림 9는 SDL의 구문으로 작성한 시나리오의 한 예이다.

```
#OBJECT
    A: F(somevideo) M(video) P(6)
    B: F(sometext) M(text) P(5)
    C: F(somesong) M(audio) P(4)
    D: F(somevoice) M(audio) P(7)
#END
#RELATION
    Parallel( Sequential( T(10),
                A(80), T(10) ),
    Sequential( T(20), B(60), T(20) ),
    C(100),
    Sequential( T(20), D(70), T(10) ) )
#END
```

그림 9. SDL을 이용하여 기술된 시나리오 예

$\Rightarrow$ Sequential(A(δ_A),T(δ),B(δ_b))

$\Rightarrow$ Sequential(A(δ_A),B(δ_b))

$\Rightarrow$ Parallel(Sequential(A(δ_A),T(δ_1)),
Sequential(T(δ_2),B(δ_B)))

$\Rightarrow$ Parallel(Sequential(T(δ_1),A(δ_A),
T(δ_2)), B(δ_B)))

$\Rightarrow$ Parallel(Sequential(A(δ_A),T(δ)),
B(δ_B))

$\Rightarrow$ Parallel(Sequential(T(δ),A(δ_A)),
B(δ_B))

$\Rightarrow$ Parallel(A(δ_A),B(δ_B))

그림 8. Sequential과 Parallel을 이용한 13 가지 시간 관계 표현

3.3. 내부 구조

본 논문에서는 동기화 과정에서 시나리오를 구성하는 객체의 특성과 객체 간의 시간 관계를 쉽게 검색할 수 있도록 우선 순위 구간(priority unit)을 기반으로한 시나리오 내부 구조를 제공한다.

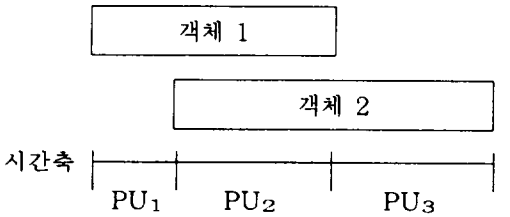

그림 10. 각 객체의 시작점과 끝점에 의해 생성되는 우선 순위 구간

우선 순위 구간이란 그림 10에 보인 것처럼 시간 축을 각 객체의 프리젠테이션 시작점과 끝점에 대해 수직으로 잘랐을 때의 한 구간을 말한다. 이 우선 순위 구간은 크게 두 가지 의미를 지니는데, 하나는 동시에 프리젠테이션될 객체의 수가 우선 순위 구간에 따라 달라진다는 것이다. 그림 10에서 보는것 처럼 우선 순위 구간이 PU1, PU2, PU3로 변함에 따라 해당 구간에 존재하는 객체의 수가 1, 2, 1로 변한다. 이는 우선 순위 구간이 각 객체의 프리젠테이션 시작점과 끝점을 기준으로 결정됐기 때문에, 어쩌면 당연한 결과라 할 것이다. 다른 하나는 각 객체에 부여된 우선 순위의 상대적 위치가 우선 순위 구간에 따라 변한다는 것이다. 그림 10에서 객체 1의 우선 순위가 객체 2 보다 높은 경우에 PU1과 PU2에서는 물론 객체 1의 우선 순위가 가장 높지만, PU3에서는 객체 2의 우선 순위가 제일 높게된다. 물론 그림 10은 극단적인 예를 보이고 있지만 이를 바탕으로 우선 순위의 상대적 위치가 변한다는 것은 쉽게 알 수 있다. 그림 11은 우선 순위 구간을 이용한 시나리오의 내부 구조를 나타낸다.

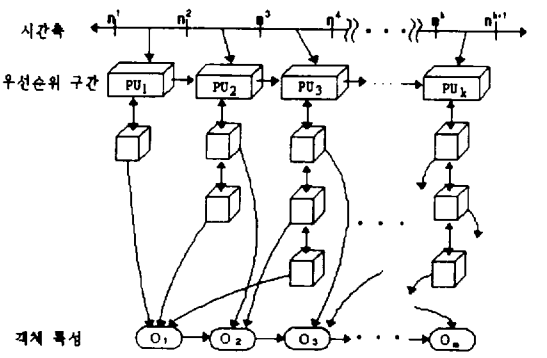

그림 11. 우선 순위 구간을 이용한 시나리오의 내부 구조

시나리오를 구성하는 각 객체의 프리젠테이션 시작점과 끝점의 집합을 오름 순으로 정렬한 결과를 n_1, n_2, n_3, ..., n_k라 할때, $PU_i(1 \le i \le k)$는 길이가 $(n_{i+1} - n_i)$인 i번째 구간을 의미한다. 하나의 우선 순위 구간에는 0개 이상의 객체가 존재하며, 모두 우선 순위에 의해 정렬되어 있다. 모든 객체는 하나 이상의 우선 순위 구간에 속하며, 이들은 동기화 과정에 필요한 객체의 특성을 공유하고 있다. 객체의 특성은 정적 특성, 데이터 화일 그리고 우선 순위 정보로 구성된다. 프리젠테이션 기간은 포함되지 않는데, 이 값은 객체가 속한 우선 순위 구간들의 길이의 합에 의해 구할 수 있으므로 별도로 저장할 필요는 없다.

그림 11에서 객체 O_1은 PU_1에서 PU_3까지 $(D_1 + D_2 + D_3)$동안 프리젠테이션되며, 객체 O_2는 PU_2부터 PU_3까지 $(D_2 + D_3)$동안 프리젠테이션됨을 알 수 있다. 이때, 각 우선 순위 구간에 존재하는 객체는 동기화 과정에서 회생자를 빨리 찾기 위해 우선 순위가 낮은 객체일수록 PU에 가깝게 정렬되어 있다.

4. 동기화 기법

멀티미디어 시스템은 여러 종류의 미디어를 통합하여 다루기 때문에 이들간의 동기화는 매우 중요한 연구 과제이다. 멀티미디어 시스템에서의 동기화란 멀티미디어 객체들을 시나리오에 기술된 시간 관계에 따라 정확한 시간에 프리젠테이션 시키기 위한 기법을 말한다. 본 논문에서 제안한 동기화 기법은 오프라인 스케쥴러에 의해 수행된다. 오프라인 스케쥴러는 프리젠테이션 수행 이전에 모든 동기화 작업을 끝마치는 일종의 정적(static) 스케쥴러로, 시나리오의 수행 가능성을 미리 판단할 수 있고, 시스템 자원을 효율적으로 이용하는 장점이 있다. 먼저 설계 과정에서 고려된 사항에 대해 살펴본 후, 오프라인 스케쥴러에 대해 살펴보기로 한다.

4.1. 요구 조건

4.1.1. 시스템 자원의 한계

멀티미디어 데이타는 일반적인 텍스트 위주의 데이타에 비해 크기가 크기 때문에, 처리에 필요한 시스템 자원이 많이 요구되는 특징이 있다. 멀티미디어 프리젠테이션 시스템에서의 자원이란 크게 멀티미디어 데이타를 저장하는데 필요한 버퍼와 이들 데이타를 디스크에서 버퍼로 읽어들이는데 필요한 전송 시간으로 생각할 수 있다. 그런데 멀티미디어 프리젠테이션은 이러한 시스템 자원이 충분한 특정 시스템에서만 이루어지는 것이 아니므로, 동기화 기법은 현 시스템의 자원 한계를 고려하고, 이를 효과적으로 이용할 수 있도록 설계되어야 한다.

4.1.2. 사용자의 요구

사용자가 멀티미디어 시나리오에 기술하는 내용은 단순히 객체들간에 있을 수 있는 시간 관계에 대한 것만은 아니다. 사용자는 시간 관계를 상호 객체간의 관계로 표시할 수도 있으며("A가 끝난 후 B를 시작해라"), 절대적인 시간을 줄 수도 있다.("시작된지 10초후에 C를 시작하라") 또한 특수 효과를 원할 수도 있다.(fade out, overlap,....) 따라서 시나리오에 기술된 내용을 정확히 파악하여 동기화에 이용해야 한다.

4.1.3. 매체의 특성

멀티미디어 시스템에서 조작하는 여러 종류의 매체들은 각기 그 특성을 가지고 있다. 그러므로 이를 스케쥴링 작업에 반영해야 한다. 정지 화상의 경우에는 일부 데이타가 손상되어도 그 범위가 크지 않으면 전체 윤곽을 이해하는데 큰 지장을 초래하지는 않는다. 그에 반해 오디오 데이타의 경우 150ms이상의 오차가 생길 경우 듣는 사람이 그 사실을 인식할 수 있다고 알려져 있다[13]. 또한 다른 미디어 데이타는 이보다 큰 오차를 허용하므로 그에 맞는 스케쥴링 작업이 요구된다.

4.1.4. 객체의 우선 순위

시스템 자원이 부족하여 프리젠테이션이 정상적으로 이루어지지 못할 경우에는 어쩔 수 없이 특정 객체에 대한 손실 작업을 수행하여야 한다. 이때, 만일 임의의 객체를 선정하여 손실시킨다면, 사용자가 원치않은 객체에 대한 손실이 발생할 수 있다. 본 논문에서는 이와 같은 문제를 해결하기 위해 객체의 선정 작업에서 사용자가 정의한 객체의 우선 순위를 고려도록 설계되었다.

4.2. 오프라인(off-line) 스케쥴러의 설계

동기화 과정은 스케쥴러에 의해 수행된다. 스케쥴러는 시나리오의 내부 구조를 입력으로 받아 3개의 과정을 수행한다. 매인 모듈에는 객체 분할(dividing object), 초기 배치(initial placement), 그리고 전이(migration) 모듈

이 있으며, 이들을 모두 우선 순위 구간 단위로 수행된다. 그 밖에 객체 손실(loss of object) 모듈이 있는데, 이는 전이 모듈이 실패했을때만 호출된다. 그럼 각각에 대해 좀더 자세히 살펴보기로 한다.

4.2.1. 객체 분할 모듈

프리젠테이션 과정에서 처리되는 객체의 단위가 지나치게 클 경우에는 수행 중에 어긋남이 발생하거나, 메모리 점유시간이 길어짐으로써 메모리를 효과적으로 활용하지 못하는 문제점이 있다. 이러한 문제점을 해결하기 위해 객체 분할 모듈에서는 현 우선 순위 구간에 속하는 객체들을 여러개의 부객체로 분할한다. 분할된 부객체의 크기는 미디어별 최소 조작 단위와 디스크 입출력 양을 고려하여 결정된다.

예를 들어 비디오 객체인 경우에는 정수개의 프레임이, 오디오인 경우에는 디스크 블럭이, 그리고 텍스트인 경우에는 전체 객체가 최소 조작 단위로 이용될 수 있다. 이러한 최소 조작 단위는 사용자가 기술한 시나리오를 읽어들이는 과정에서 각 객체의 정적 특성을 이용하여 결정된다. 그림 12는 최소 조작 단위로 분할된 부객체의 구조를 보이고 있다.

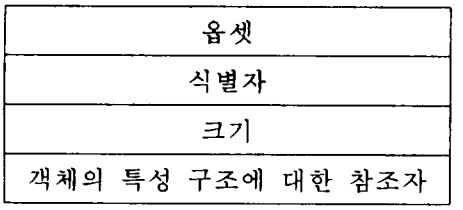

그림 12. 최소 조작 단위로 분할된 부객체의 구성 요소

4.2.2. 초기 배치 모듈

초기 배치 모듈에서는 우선 순위 구간 내에 존재하는 부객체들을 정확한 시점에 프리젠테이션하기 위해 각 부객체를 그림 13과 같이 선반입(prefetching), 시작(start), 종료(end)의 3 개의 스케쥴링 객체로 분할한다. 선반입 객체는 부객체를 버퍼로 읽어들이기 위한 작업을 나타내는 객체이며, 시작 및 종료 객체는 각각 부객체의 프리젠테이션 시작 및 종료 시점을 나타내는 객체이다. 각 스케쥴링 객체에는 시작 시간과 지속시간(duration)이 있는데, 이들은 다른 부객체와 상관없이, 해당 부객체의 프리젠테이션 시점에 맞도록 초기화된다. 예를들어 부객체 A의 프리젠테이션 시작 시간과 지속시간이 각각 10과 5이고, 디스크에서 버퍼로 읽어들이는데 걸리는 시간이 3이라면, 부객체 A에 대한 선반입 객체는 시작이 7, 지속시간이 3으로, 시작 객체는 시작이 10, 지속시간이 0으로, 종료 객체는 시작이 15, 지속시간이 0으로 각각 초기화된다. 이와 같이 분할된 스케쥴링 객체들은 순서에 관계없이 전체가 하나의 리스트를 구성한다.

그림 13. 하나의 부객체에 대응되는 3 개의 스케쥴링 객체

4.2.3. 전이 모듈

초기 배치의 결과로 생기는 리스트에는 시간이 겹치는 스케쥴링 객체들이 존재할 수 있다. 전이 모듈에서는 이러한 객체들을 없애기 위해 스케쥴링 객체들 중 선반입 객체를 리스트상의 빈 구간으로 이동시킨다. 물론 새로운 선반입 객체의 위치는 그 이전 위치보다 앞쪽에 있어야 한다. 이와 달리 시작 객체와 종료 객체는 정해진 시간에 수행되어야 정확한 동기화를 이룰 수 있으므로 전이 대상이 될 수

없다. 전이가 발생하면 전이된 객체가 리스트 상의 앞 객체와 겹치는 지를 추가로 검사하고, 만일 충돌이 발생하면 더이상 충돌이 없거나, 리스트의 제일 앞으로 갈때까지 위의 과정을 반복하게 된다.

모든 전이 과정은 사용 가능한 버퍼의 크기를 고려하여 이루어진다. 가용 버퍼의 크기는 스케쥴링 객체내에 기록되어 다음 스케쥴링 객체에 전달된다. 따라서 만일 앞의 스케쥴링 객체로부터 충분한 크기의 버퍼를 전달받지 못하면 전이 과정이 실패한 것이되며, 결과적으로 객체 손실 모듈이 호출된다. 모든 객체에 대한 전이 작업이 버퍼 공간 부족없이 완료되면 각 스케쥴링 객체는 작업 시작 시간을 기준으로 정렬된 형태가 되며, 이 리스트가 결국 스케쥴링 작업의 결과가 된다.

4.2.4. 객체 손실 모듈

전이 모듈의 수행이 완료되면 서로 중복되는 시간 구간에 대한 처리는 모두 해결되지만, 버퍼 공간이 부족한 경우에 대한 처리는 이루어지지 않는다. 객체 손실 모듈은 버퍼 부족이 발생한 경우에만 호출되며, 우선 순위 구간 내에 있는 객체들 중 우선 순위가 가장 갖는 객체를 희생자로 선택하여 손실시킴으로써 이 문제를 해결한다. 이때, 우선 순위 구간 내에 존재하는 객체들은 우선 순위가 낮은 것 부터 정렬되어 있으므로 희생자를 찾는 작업은 쉽게 수행될 수 있다. 선정된 희생자에 대한 손실 작업은 사용자가 정의한 우선 순위 테이블에 따라 이루어진다. 우선 순위 테이블은 객체의 미디어 종류와 우선 순위에 따라 어느 정도의 손실 작업이 이루어져야 하는가에 대한 정보를 가지고 있다. 손실의 정도는 객체의 일부만을 희생하는 것부터 전체를 프리젠테이션 대상에서 제외하는 등의 여러 단계로 나뉘어질 수 있다. 손실 작업이 완료되면 스케쥴러는 현 우선 순위 구간에 대한 스케쥴링 작업을 성공할때 까지 계속 수행한다.

5. 구 현

5.1. SDL

SDL 부분에서는 사용자가 기술한 시나리오를 읽어서 문법을 검사한 후, 그림 14의 자료구조를 이용하여 멀티미디어 객체의 속성 구조와 시나리오의 내부 구조를 생성한다. 그림 14에서 OBJ는 멀티미디어 객체의 속성을, PU는 우선 순위 구간을, 그리고 PUOBJ는 우선 순위 구간에 속하는 멀티미디어 객체를 각각 나타낸다. 시나리오의 내부 구조는 PU의 리스트로 표현되며, 각 PU는 PUOBJ의 리스트를 포함하고 있다. OBJ 가운에 margin은 분할한 부객체가 우선 순위 구간을 넘어갈 경우, 다음 구간에 대한 해당 객체의 시작 시간을 조정하기 위해 사용된다. min_size와 ptime은 각각 해당 객체를 부객체로 분할했을때, 부객체의 크기와, 프리젠테이션 시간을 나타내며, 데이타 화일의 헤더에 있는 정적 특성을 읽어서 결정한다. 현재 비디오의 경우에는 한 프레임, 오디오의 경우에는 한 디스크 블럭, 이미지와 텍스트인 경우에는 전체 데이타 크기를 최소 단위로 설정하였다.

5.2. 동기화

동기화 부분은 그림 15와 같이 각 우선 순위 구간별로 객체 분할, 초기 배치, 그리고 전이 과정을 수행하며, 전이 과정이 실패할 경우에는 해당 구간에 대한 객체 손실 과정을 수행한다.

5.2.1. 객체 분할

객체 분할 과정은 우선 순위 구간 내에 존재하는 멀티미디어 객체들을 그림 16의 SUBOBJ

```
typedef struct object {
        char    name[20];   /* Object Name */
        char    file[20];   /* Data File */
        int     fp;         /* File Descriptor */
        int     priority;   /* Priority of Multimedia Object */
        int     mtype;      /* Media Type */
        int     start;      /* Presentation Start Time */
        int     finish;     /* Presentation End Time */
/* The followings are used at synchronization time */
        int     min_size;   /* Minimum Processing Data Size (Bytes) */
        int     ptime;      /* Presentation Time for Min Proc. Data */
        int     margin;     /* The Next Subobject Start after this time */
        int     id;
        int     offset;     /* Offset of Data File */
        struct  object* next;
        struct  object* prev;
} OBJ;

typedef struct puobj {
        OBJ*    obj;        /* Attributes of multimedia object */
        struct  puobj* next;
        struct  puobj* prev;
} PUOBJ;

typedef struct pu {
        int     duration;   /* Duration of current PU */
        PUOBJ*  puobj;      /* First subobject in current PU */
        struct pu* next;
} PU;
```

그림 14. 시나리오 기술 언어 부분에서 사용하는 자료구조

```
sync()
{
     pu = 첫번째 우선 순위 구간
     while (pu가 마지막 우선 순위 구간이 아님) {
             pu에 있는 멀티미디어 객체를 부객체로 분할
             부객체를 스케쥴링 객체로 분할하여 초기 배치 시킴
             while (migration() = 버퍼부족)
                     pu에 대한 객체 손실 작업
             pu = pu->next
     }
}
```

그림 15. 동기화 과정 알고리즘

```
typedef struct {                        /* Sub-object */
        int             offset;         /* Offset of Data File */
        int             id;             /* Subobject id */
        unsigned int    size;           /* Size of subobject */
        OBJ*            obj;    /* Attributes of multimedia object */
} SUBOBJ;

typedef struct sobj {                   /* Scheduling Object */
        int             start;          /* Start time */
        int             dur;            /* Duration */
        char            type;           /* FETCH, START, END */
        int             memory;         /* Available Memory Size */
        struct sobj*    next;
        struct sobj*    prev;
        SUBOBJ*         subobj;
} SOBJ;
```

그림 16. 동기화 부분에서 사용하는 자료구조

```
migration()
{
     sobj = 리스트 상의 첫번째 스케쥴링 객체
     while (sobj->next가 마지막 스케쥴링 객체가 아님) {
            if (sobj와 sobj->next가 겹치고 두 객체의 type이 모두 FETCH
가 아닌 경우) {       두 스케쥴링 객체의 위치를 바꿈
                      sobj = sobj->prev->prev
                      continue
            }
            if (sobj와 sobj->next가 겹침) {
                      sobj의 시작 = sobj->next의 시작 - sobj의 기간
                      sobj = sobj->prev
            }
            else {    if (버퍼가 부족)
                              return 버퍼부족
                      가용 버퍼의 크기를 변경
                      sobj = sobj->next
            }
     }
     return 전이완료
}
```

그림 17. 시간이 겹치는 스케쥴링 객체를 해결하는 전이 과정 알고리즘

구조를 갖는 여러개의 부객체로 분할한다. 이 때 부객체의 크기는 멀티미디어 객체의 속성 구조에 명시되어 있는 min_size와 ptime에 의해 결정된다.

5.2.2. 초기 배치

초기 배치 과정에서는 각 부객체를 3 개의 스케쥴링 객체로 나눈후, 이들로 구성되는 리스트를 생성한다. 스케쥴링 객체는 그림 16의 SOBJ 구조를 가지며, type에 따라 FETCH, START, END로 구분된다. SOBJ에서 start와 dur은 부객체의 프리젠테이션 시점에 맞도록 초기화되며, memory 현재 가용한 버퍼의 크기를 검사하는데 사용된다. 이때, 리스트 상의 맨 처음 스케쥴링 객체는 시스템에서 현재 사용할 수 있는 최대의 버퍼 크기를 초기값으로 기록하며, 나머지 객체들은 앞에 있는 객체의 memory 값에서 자신이 필요한 버퍼의 크기를 뺀 값을 기록함으로써, 뒤에 있는 스케쥴링 객체가 현재 사용 가능한 버퍼의 크기가 얼마인지를 쉽게 알 수 있도록 한다. 초기 배치 과정이 끝나게 되면 스케쥴링에 필요한 자료구조의 생성은 완료된다.

5.2.3. 전이

전이 과정은 동기화의 가장 주된 루틴으로, 초기 배치의 결과로 생성되는 리스트를 검사하여 시간이 겹치는 스케쥴링 객체를 앞으로 전이시키는 과정을 수행한다. 단, FETCH 객체만이 앞으로 전이될 수 있으며, START와 END 객체는 전이되지 않는다. migration은 그림 17과 같이 수행된다.

그림 17에서 FETCH 객체들의 초기 순서를 변경시키지 않기위해 시간 겹침 검사를 두군데서 하는데, 그 이유는 될수있는대로 프리젠테이션 시점에 가까이 FETCH 객체를 두는것이 버퍼이용면에서 효율적이기 때문이다.

5.2.4. 객체 손실

객체 손실 작업은 멀티미디어 객체의 미디어 종류와 우선 순위에 따라 객체 전체를 프리젠테이션 대상에서 완전히 제외시키는 것에서 부터 객체의 일부를 제외시키는 것이 이르기까지 여러가지가 있으며, 이는 그림 18과 같은 우선 순위 테이블에 명시되어 있다. 우선 순위 테이블은 시스템 환경에 따라 사용자가 원하는대로 변경할 수 있다. 그림 18에서

<table>
<tr><th rowspan="2">우선순위</th><th colspan="5">미디어 종류</th></tr>
<tr><th>오디오</th><th>비디오</th><th>이미지</th><th>텍스트</th><th>그래픽</th></tr>
<tr><td>0</td><td colspan="5">무조건 없앰</td></tr>
<tr><td>1</td><td colspan="5">충돌이 발생하면 없앰</td></tr>
<tr><td rowspan="3">2
3

4
5
6

7
8
9</td><td rowspan="2">충돌이
발생하면없앰</td><td>정지화면으로
대체</td><td rowspan="3">충돌이
발생하면없앰</td><td rowspan="3">충돌이
발생하면
없앰</td><td rowspan="3">충돌이
발생하면
없앰</td></tr>
<tr><td>우선 순위 구간
내에 있는 객체
를 없앰</td></tr>
<tr><td>우선 순위 구간
내에 있는 객체
를 없앰</td><td>우선 순위 구간
내의 일부 프레
임을 누락</td></tr>
<tr><td>10</td><td colspan="5">절대로 회생되어서는 안됨</td></tr>
</table>

그림 18. 객체 손실 작업을 위한 우선 순위 테이블

우선 순위 0과 10은 특별한 용도로 사용되는데, 0은 어떤 멀티미디어 객체의 데이타가 파손되어 해당 객체를 시나리오에서 완전히 제외시킬 경우에, 10은 절대로 회생되어서는 않되는 객체에 사용된다. 만일 우선 순위가 10인 객체를 회생해야 하는 경우에는 동기화가 실패했음을 사용자에게 알린다.

모든 우선 순위 구간에 대해 전이 과정이 완료되면, 스케쥴링 객체의 리스트는 프리젠테이션 엔진에 전달되어 실제 프리젠테이션이 수행된다. 프리젠테이션 엔진은 스케쥴링 객체의 리스트를 따라가면서 객체의 타입에 따라 디스크로부터 데이타를 읽거나 특정 출력 장치에 신호를 보내는 역할을 수행하는 하위 수준의 모듈이다. 이부분은 본 논문의 범위에 포함되지 않으므로 자세한 설명은 생략한다.

6. 결 론

멀티미디어 프리젠테이션 시스템은 크기가 큰 데이타들을 다루기 때문에 자원이 충분치 않은 시스템 환경에서는 미디어-간에 어긋남이 발생하는 문제점이 있다. 기존의 연구에서는 동적인 누락이나 중복 기법을 이용하여 어긋남을 해결하고 있으나, 이 방법은 예상치 못한 데이타의 손실을 유발할 수 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 객체의 손실을 허용하는 멀티미디어 프리젠테이션 시스템을 제안하였으며, 이를 구성하는 SDL과 동기화 기법을 설계, 구현하였다. 제안된 시스템은 우선 순위가 낮은 객체를 회생시킴으로써, 어긋남이 발생하지 않고 한정된 시스템 자원내에서 수행가능한 프리젠테이션을 제공한다.

SDL은 객체간의 시간 관계와 우선 순위를 표현하는데 이용된다. SDL로 작성된 시나리오는 내부적으로 우선 순위 구간을 이용하여 저장되며, 이는 동기화 과정에서 객체 간의 시간 관계와 각 객체의 특성을 쉽게 파악하는데 이용된다.

오프라인 스케쥴러에 의해 수행되는 동기화 작업은 시나리오의 수행 가능성을 미리 예측할 수 있는 장점이 있으며, 버퍼의 효율적 이용과 정확한 동기화를 위해 각 객체를 최소의 단위로 분할하여 처리한다. 우선 순위 구간별로 수행되는 프리젠테이션 가능성 검사에서는 자원이 부족한 경우 우선 순위가 낮은 객체를 회생시킴으로써 미디어 객체간의 어긋남을 사전에 해결한다.

본 논문에서 사용하는 동기화 기법은 자원 관리에 기반을 두고 있으므로 응용 프로그램 수준에서 수행하는데는 한계가 있다. 이는 응용 프로그램 수준에서는 효과적인 자원 관리가 불가능하기 때문이며, 결국 멀티미디어 운영체제의 필요성이 대두된다. 본 논문에서 구현한 동기화 기법은 앞으로 멀티미디어 운영체제의 자원 할당 정책으로 이용될 수 있을 것이다.

참고 문헌

[1] E. Fiume, D. Tsichritzis, and L. Dami, "A Temporal Scripting Language for Object-Oriented Animation", Proceedings of Eurographics 1987(North-Holland), Elsevier Science Publishers, Amsterdam, 1987.

[2] David P. Anderson and Geroge Homsy, "A continuous Media I/O Server and Its Synchronization Mechanism", COMPUTER, Oct. 1991, pp. 51-57.

[3] T. D. C. Little and F. Kao, "An Intermedia Skew Control System for Multimedia Data Presentation", 3rd International workshop on Network and Operating System support for

Digital Audio and Video, Nov. 1992, pp 121-132.

[4] Ralf Steinmetz, "Synchronization Properties in Multimedia Systems", Journal on Selected Areas in Communications, Vol. 8, No. 3, Apr. 1990, pp. 401-412.

[5] R. B. Dannenberg, T. Neuendorffer, J. M. Newcomer, and D. Rubine, "Tactus: Toolkit-Level Support for Synchronized Interactive Meultimedia", 3rd International workshop on Network and Operating System support for Digital Audio and Video, Nov. 1992, pp. 264-275.

[6] T. D. C. Little and A. Ghafoor, "Multimedia Object Models for Synchronization and Databases", 6th International Conference on Data Engineering, 1990, pp. 20-27.

[7] D. J. Moore, "Multimedia Presentation development using the Audio Visual Connection", IBM System Journal, Vol. 29, No. 4, 1990, pp 494-508.

[8] Rei Hamakawa and Jun Rekimoto, "Object Composition and Playback Models for Handling Multimedia Data", First ACM International Conference on Multimedia, 1993, pp 273-281.

[9] James F. Allen, "Maintaining Knowledge about Temporal Intervals", Communications of the ACM, Vol. 26, No. 11, 1983, pp 832-843.

[10] Petra Hoepner, "Synchronizing the Presentation of Multimedia Objects - ODA Extensions -", SIGOIS Bulletin, Vol. 12, No. 1, July 1991, pp 19-32.

[11] P. Venkat Rangan and Harrick M. Vin, "Designing File Systems for Digital Video and Audio", 13th ACM Symposium on Operating System Principles, Operating Systems Review, Vol. 25, No. 5, Oct. 1991, pp 69-79.

[12] 홍명희, 김우생, "멀티미디어 데이타 동기화를 위한 스케쥴링 기법", 데이타베이스연구회지, 8권, 1993, pp 35-50.

[13] T. D. C. Little and A. Ghafoor, "Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks", Computer, Oct., 1991, pp 42-50.