효율적인 멀티미디어 문서 생성을 위한 통합 저작 시스템의 설계 및 구현

(Design and Implementation of an Integrated Authoring System for Efficient Multimedia Document Generation)†

최 숙 영*, 신 현 산**, 유 관 종***
(Sook-Young Choi) (Hyun-San Sin) (Kwan-Jong Yoo)

요 약 본 논문에서는 쉽고 융통성 있는 저작 시스템 개발을 목표로 다음과 같은 특징을 갖는 시스템을 구현하였다. 첫째, 사용자가 멀티미디어 문서를 쉽게 저작하고 프리젠테이션 할 수 있는 시각 환경을 제공하며, 실행중에 발생되는 미디어 실행시간 변경을 효과적으로 처리할 수 있는 인과성 시간 관계를 정의한다. 둘째, 저작 내용을 트리 형태의 내부 구조로 저장함으로써 효과적인 문서 파싱을 수행하며, 실행 시간의 효과적인 프리젠테이션과 사용자 상호작용을 처리하기 위한 프리젠테이션 엔진을 지원한다.

Abstract In this paper, we implemented an authoring system in which it's easy and flexible for user to author multimedia documents. It can be characterized by the following properties. First, it provides a visual interface for easy multimedia authorings and presentations and defines a causal relation that can manage changes of media duration. Second, it provides an internal representation using tree and thus supports an effective document parsing. It also supports a presentation engine for effective presentations and user interactions in run-time.

1. 서 론

멀티미디어 저작이란 텍스트, 그래픽, 이미지, 비디오, 오디오, 애니메이션 등의 다양한 미디어들을 사용자와 컴퓨터간에 서로 대화하면서 각 미디어의 실행 순서를 정의하는 일련의 편집 작업을 의미한다[1,2,3]

멀티미디어의 대중화와 함께 멀티미디어 시나리오 저작이나 타이틀 제작을 전문업체에 의뢰하기 보다 자신이 직접 제작할 필요성이 많아졌다. 응용 분야에 따라서는 프로그램 전문가보다도 그 역할 면에서나 능력 면에서 더욱 전문성을 발휘할 수도 있다는 이점이 있다. 그러나 이러한 저작 및 프리젠테이션 시스템을 사용하여 저작하기에는 그 저작 환경이 어렵고 복잡하며, 아울러 저작에 따른 자동화된 멀티미디어 데이터 처리 환경이 미비한 실정이다. 기존의 시스템이 갖는 문제점을 사용자 저작 환경과 멀티미디어 데이터 처리 환경 측면에서 정리하면 다음과 같다[4].

사용자 저작 환경 측면에서는 다양한 멀티미디어 응용 분야 확대 측면에서 시각적 저작 환경 제공을 위한 연구가 미흡하다고 볼 수 있다. 기존에 개발된 저작 시스템들은 주로 스크립트를 기반으로 하는 프로그래밍 언어를 사용하기 때문에 초보자들이 배우기도 어렵고 사용 방법이 까다롭다. 또한 저작시 발생되는 일관성 오류를 즉시 사용자에게 알려주는 동적 저작(dynamic authoring) 기능이 미흡하다. 동적 저작은 편집 단계에서부터 실행 단계까지 일관성 있는 프로그램이 되도록 하는 저작 시스템의 중요한 기능이다. 멀티미디어 저작 환경과 함께 데이터 처리

†본 논문은 1999년도 우석대학교 학술연구비 지원을 받았음
*우석대학교 컴퓨터교육과 조교수
**충청대학 전산학과 조교수
***충남대학교 컴퓨터과학과 교수

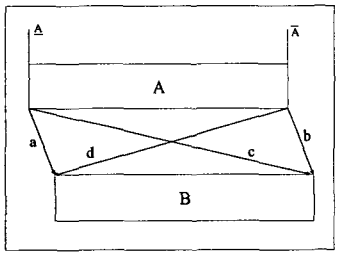
환경 측면에서는 융통성 있는 프리젠테이션 엔진 연구가 미흡하다. 프리젠테이션 엔진은 미디어의 수행시간이 실시간에 바뀌어도 동기가 깨지지 않도록 융통성을 지원할 수 있어야 하며, 아울러 사용자 상호작용을 통한 프리젠테이션 수행시간 조절을 지원하며, 이를 통합적으로 처리하는 동기 및 비동기 메커니즘이 구축되어야 할 것이다.

따라서 본 연구에서는 위와 같은 기존 시스템의 저작 환경과 데이터 처리 환경에서의 문제점을 개선하기 위하여 동적 저작 모델링을 제안하고 있으며, 이를 위해 기존의 Allen[5]이 제시한 미디어간 시간 관계를 그대로 적용하지 않고 인과 관계(causal relation)를 바탕으로 새로운 시간 관계를 정의하였다. 또한, 이러한 모델을 지원할 수 있는 저작 시스템을 구현하였으며, 그 시스템은 시각 인터페이스, 파싱 단계, 프리젠테이션 엔진 부분으로 구성되고, 프리젠테이션에 참여하는 미디어간의 동기 및 사용자 상호작용에 의한 비동기를 효율적으로 지원하고 있다.

2. 인과성 시관 관계

본 연구에서 제안하는 저작 시스템은 인과성 관계를 바탕으로 시간 관계를 명세하고, 동기화를 위해 참조점(reference point) 방식을 사용한다. 본 장에서는 이 시간 관계 모델에 대해 간략하게 기술하며, 이에 대한 자세한 설명은 [11]에서 참조될 수 있다.

인과성 관계는 이유, 목적, 양보, 의뢰로써[6], 프리젠테이션 상에 실행되는 미디어들은 서로간에 어떤 이유나 목적에 의해 서로 관련성을 맺게 된다.

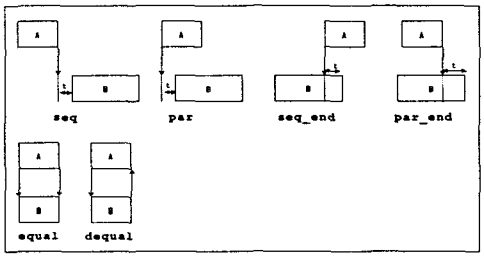


<그림 1> 객체간의 인과성 관계

인과성을 표현하는 시간 관계는 하나의 객체(A)에 존재하는 두 개의 동기점인 시작점(A), 끝점($\bar{A}$)을 사용해서 크게 4가지로 구성할 수 있다. 시작점은 한 객체의 시작을 나타내고, 끝점은 한 객체의 멈춤을 나타낸다.

<그림 1>은 두 개의 객체 A, B 간에 존재하는 인과성 관계를 표현한 것이다. A, B 각 동기점에 의해서 한 객체는 cause 객체가 되고, 다른 객체는 effect 객체가 된다. 이런 관계는 화살표 방향에 따라 그 역할이 결정된다. 즉 cause 이벤트를 발생하는 A 객체는 능동 객체의 역할을 수행하고, effect 이벤트가 발생하는 다른 한 객체는 수동 객체의 역할을 수행하는 것이다. 4가지 관계를 설명하면, a는 A가 시작하면서 B를 시작시키는 관계, b는 A가 끝나면서 B를 끝내는 관계, c는 A가 시작하면서 B를 끝내는 관계, 마지막으로 d는 A가 끝나면서 B를 시작시키는 관계를 나타낸다.

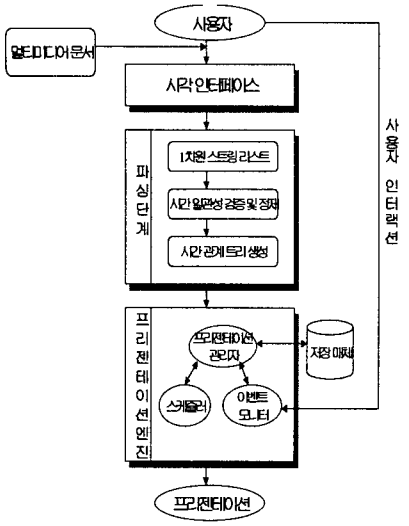
<그림 1>를 바탕으로 본 연구에서 정의한 인과성 기반 시간 관계는 <그림 2>에 나타냈다. 기본적인 4가지 인과성 관계와 지연시간을 포함하여 seq, par, seq_end, par_end를 정의했고, 추가적으로 동일성(equality) 관계를 equal, dequal로 정의했다.



<그림 2> 인과성 시간 관계

3. 저작 시스템 설계

저작 시스템 설계는 앞 절의 시간 관계 모델을 기반으로 했다. 사용자 인터페이스는 그래픽 사용자 인터페이스 환경을 위하여, 아이콘 중심 프로그래밍을 할 수 있는 시각 인터페이스를 설계했고, 파싱은 서로 다른 특성을 갖는 미디어들을 통합 처리할 수 있는 공통된 자료 구조를 사용하여 내부 구조를 단계별로 설계했으며, 파싱 순서는 1차원 스트링 리스트 생성, 문서 규격 및 일관성 검증, 시간 트리 생성 등 3 단계로 수행했다. 프리젠테이션 엔진에서는 생성된 최종 내부 구조인 복합 트리를 동기와 비동기를 고려하여 스케줄하고 프리젠테이션을 수행하도록 설계하였다. 이러한 저작 모델의 처리 흐름을 나타낸 시스템 구성은 <그림 3>과 같다.



<그림 3> 시스템 구성도

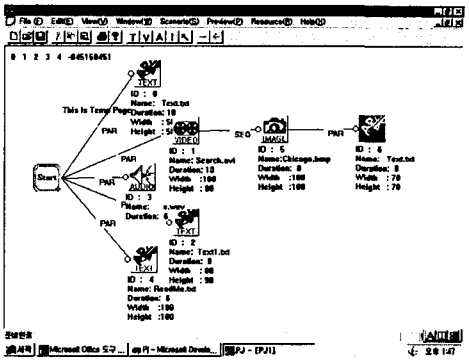
3.1 사용자 인터페이스

사용자 인터페이스는 시간 뷰(view), 상호작용 뷰를 제공했다. 시간 뷰는 프리젠테이션에 참여하는 미디어들의 실행 시간에 대한 시나리오를 아이콘을 사용하여 프로그램하는 인터페이스이다. 상호작용 뷰는 작성된 시간 뷰를 프리젠테이션 화면에 실행하면서 사용자의 의도를 시나리오 상에 반영하는 인터페이스이다.

1) 시간 뷰

시간 뷰 저작 화면에서는 여러 가지 미디어 객체 아이콘을 사용해서 객체간의 시간 관계를 정의하고, 정의된 각 미디어 객체들의 미디어 특성 값과 시간 관계 특성 값을 입력한다. 시간 뷰 저작 화면은 미디어 객체 아이콘과 시간 관계를 주는 아이콘으로 구성되며 이는 툴바를 통해 제공된다. 미디어 객체의 종류는 텍스트, 비디오, 오디오, 이미지, 애니메이션, 하이퍼링크 등을 비롯한 여러 가지가 제공된다. <그림 4>는 시간 뷰를 보여주고 있다.

시간 관계를 연결시켜 주는 아이콘(↖:시간 관계 아이콘)은 미디어 객체간에 인과성 시간 관계를 줄 때 사용하는 것으로, 이 아이콘을 사용하여 인과성 시간 관계가 있는 두 개의 미디어를 연결시킨다. 시스템에서 정의된 시간 관계의 종류로는 seq, par, seq_end, par_end, equal, dequal 등이 있다.



<그림 4> 시간 뷰

2) 상호작용 뷰

시나리오 저작이 완료되면 상호작용 뷰를 통하여 프리젠테이션 화면에 그 결과를 실행하면서 사용자가 시스템과 상호작용을 수행한다. 상호작용 뷰에서 사용자가 원하는 부분을 지정하여 실행시키거나, 또는 수행시간 값을 입력하여 전체적인 프리젠테이션 수행시간을 조절한다. 전체 프리젠테이션을 대상으로 수행되는 상호작용 기능을 다음과 같이 설계했다. 실행 시작, 미디어 건너뛰기, 실행 일시 정지, 실행 멈춤, 실행 속도 조절 등이 있다.

3. 2 파싱

파싱은 시각 인터페이스를 사용하여 작성된 시나리오를 스캐닝을 통해 1차원 스트링 리스트 생성 단계, 1차원 스트링 리스트의 시간 일관성 검증 및 정제 단계, 정제된 1차원 리스트를 시간 이진 트리로 변환하는 단계로 구성된다.

1) 1차원 스트링 리스트 생성 단계

시나리오의 구성은 <그림 4>와 같이 그래프 형태이기 때문에 깊이 우선 탐색(Depth First Search) 방법으로 1차원 리스트로 변환한다.

2) 시간의 일관성 검증 및 정제 단계

프리젠테이션 미디어간의 시간 일관성 검증은, 객체간에 발생될 수 있는 시간 관계가 기존의 시간 관계에 영향을 주는지 안주는 지를 판별하는 것이다. 시간 일관성을 검증하는 방법은 미디어간에 가능한 파생 시간 관계를 비교하면 된다. 여기서 파생 시간 관계란 객체간의 삼각 관계에 의해서 발생될 수 있는 시간 관계를 말하는 것으로, 새로 저작하는 미디어 객체가 아직 다른 미디어와 관계가

없거나, 또는 발생 가능한 삼각 관계에 포함될 때 시간 일관성은 유지되는 것이다.

본 논문에서는 시간 일관성 검증을 위해서 Allen이 사용한 파생시간 관계표 의미를 본 논문의 인과성 시간 관계에 의거하여 나타낸다. 검증 방법은 편집 단계에서 생성된 미디어간 파생시간 관계를 이용하여 파생 시간을 비교한다.

3) 이진트리 생성 단계

정제된 1차원 스트링 리스트는 이진 트리 형태로 변환한다. 1차원 리스트가 구조화 형태이기 때문에 이를 그대로 이진트리로 변환한다.

이 트리의 2가지 노드가 갖는 정보는 비단말 노드는 시간 관계 정보와 2개의 자식 객체 포인터를 갖고, 단말 노드는 미디어 객체 종류, 객체가 저장되어 있는 실제 주소 값, 그리고 시간과 공간에 관련된 각 미디어 객체의 특성값들을 가진다.

4. 프리젠테이션 엔진

프리젠테이션 엔진은 스케쥴러(scheduler), 프리젠테이션 관리자(presentation manager), 이벤트 모니터(event monitor)로 구성되어 프리젠테이션과 사용자 상호 작용을 효과적으로 수행한다.

4.1 스케쥴러

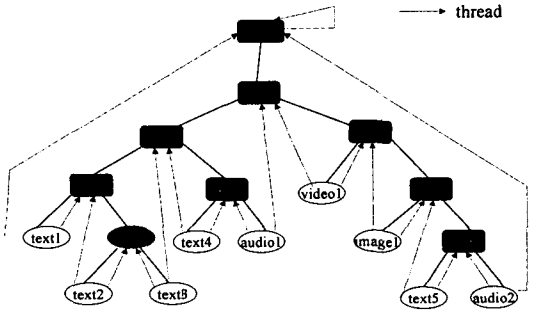
스케쥴러에서는 프리젠테이션을 위한 각 미디어들의 시작 시간과 종료 시간을 구하여 프리젠테이션 관리자에게 제공한다.

1) 프리젠테이션 시작 시간 결정

파싱 단계를 거쳐 생성된 ACT 트리를 운행하면서 프리젠테이션 시작 시간이 결정된다. 본 모델에서는 이 시작 시간을 효과적으로 계산할 수 있도록 스레드 이진 트리(threaded binary tree)를 이용하여 구현하였다. ACT 트리에서 한 미디어의 시작 시간은 상위 비단말 노드(시간 관계), 특히 중위(indorder) 방식으로 트리를 운행할 때 그 전 노드(predecessor)에 의해 영향을 받기 때문에 스레드 이진 트리[10]를 이용하여 효과적으로 구현 할 수 있다.

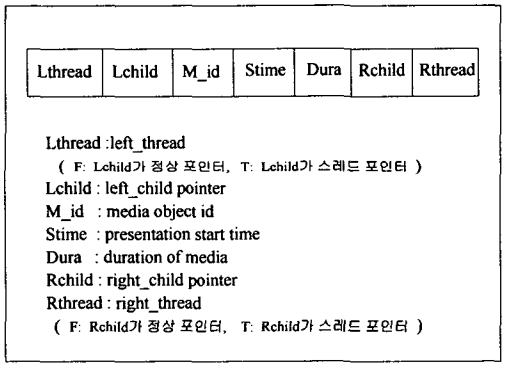
스레드 이진 트리는 널(null) 링크를 사용하여 한 노드의 그 전노드와 후노드(successor)를 가리키도록 함으

<그림 5> 스레드 이진 트리를 이용한



미디어간의 시간 관계

로서 널링크를 활용하고, 트리 운행을 효과적으로 수행할 수 있다. <그림 5>은 파싱 단계에서 생성된 ACT 트리를 스레드 이진 트리 형태로 표현한 것이다.



<그림 6> 스레드 이진 트리에서 한 단말

<그림 5>에서 video1 미디어 객체의 좌측 자식 노드 포인터는 좌측 상위 비단말 노드인 par 시간 관계로서 순회 과정에서 그 전노드이며, 우측 자식 노드 포인터는 우측 상위 비단말 노드인 seq 시간 관계로서 순회 과정에서 그 후노드를 나타내고 있다. 각 단말 노드들은 미디어 객체들을 나타내며, <그림 6>과 같은 필드들로 구성된다. Lthread와 Rthread 필드는 좌우 스레드 포인터와 정상 포인터를 구분하기 위한 필드로 Lthread가 false이면 비단말 노드로서 Lchild가 정상 포인터를 나타내며, true이면 단말노드로서 Lchild가 스레드 포인터를 나타낸다.

<그림 7>은 이 스레드 이진 트리를 이용하여 미디어들의 프리젠테이션 시작 시간을 구하는 알고리즘이다.

Procedure Pres_Stime(T)

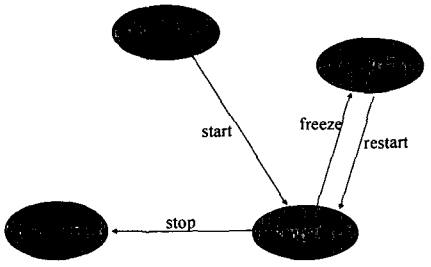
```
begin
  T := HEAD
  T := INSUC(T)
  Stime(T) := 0
  repeat
    T := INSUC(T)
    if Lthread(T) = true
       case  Lchild(T) : par
             temp := LMOST(Lchild(T))
             Stime(T) := Stime(temp) +delay
       case  Lchild(T) : seq
             temp := LMOST(Lchild(T))
             Stime(T) := Stime(temp) +
                              Dura(temp) + delay
       case  Lchild(T) : seq_end
             temp := LMOST(Lchild(T))
             Stime(T) := (Stime(temp) + delay)
                              - Dura(T)
       case  Lchild(T) : par_end
             temp := LMOST(Lchild(T))
             Stime(T) := (Stime(temp) + Dura(temp)
                              + delay) - Dura(T)
       case  Lchild(T) : equal
             temp := LMOST(Lchild(T))
             Stime(T) := Stime(temp)
       end case
  until T = HEAD
end Pres_Stime

Procedure  LMOST(X)
begin
   do
     P := Lchild(X)
   while(not Lthread(P))
   return P
end LMOST

Procedure  INSUC(X)
begin
   P := Rchild(X)
   if(not Rthread(X)) then
      while(not Lthread(P)) do
         P := Lchild(P)
   return P
end INSUC
```

<그림 7> 프리젠테이션 시작 시간 결정 알고리즘

Pres_Stime() 프로시져는 프리젠테이션 시 미디어들의 시작 시간을 구하는 알고리즘으로, 스레드 이진 트리를 중위 순회함으로서 수행된다. 모든 트리는 헤드(head) 노드의 왼쪽 서브 트리이고, INSUC(T)를 수행함으로서 트리를 순회하게 된다. INSUC(T)는 스레드 이진 트리에서 노드 T의 중위 후속자를 찾는 알고리즘이다. LMOST(T)는 그 단말노드를 찾는 알고리즘을 나타낸다. <그림 7>에서 볼 수 있는 바와 같이, 그 비단말 노드가 유지하는 시간 관계에 따라 계산 과정이 달라진다.

4.2 프리젠테이션 관리자

프리젠테이션 관리자는 스케줄러에서 제공하는 미디어들의 프리젠테이션 시작 시간과 종료 시간에 따라 미디어들을 수행시킨다. 또한 이벤트 모니터로부터 사용자 상호작용이 발생하였다는 신호를 받고 그에 따른 작업을 처리하면서 프리젠테이션을 관리한다. 프리젠테이션 과정에 따라 변화되는 객체들의 상태 정보를 유지하고 관리한다. 프리젠테이션 관리자의 처리 과정을 정리하면 다음과 같다.



<그림 8> 미디어 객체의 상태 변화

1) 실행시간에 유지되는 미디어 객체들의 상태

프리젠테이션 관리자는 실행시 미디어 객체들의 상태를 관리하며, 사용자 상호 작용이 발생될 경우 그 객체의 상태 정보를 이용하여 처리한다. 실행시 변화되는 객체들의 상태 전이도는 <그림 8>와 같다. <그림 8>과 같이 객체들의 상태는 크게 idle, ready, complete, run으로 구성되며, 실행시간의 어느 한 시점에서 이 4가지 상태 중 한 상태를 유지하게 된다. Idle은 객체의 초기 상태, ready는 지연시간 및 일시 정지를 위한 대기 상태, run은 디스플레이되는 실행 상태, complete는 객체의 수행 완료 상태를 나타낸다. 위의 상태 전이도에서 간선들은 발생되는 이벤트들을 나타내며, 각 이벤트가 발생하는 경우 객체들의 상태 변화를 볼 수 있다.

2) 프리젠테이션 관리자의 처리 과정

가) 사용자로부터 프리젠테이션 시작 명령을 받고 프리젠테이션 관리자는 스케줄러에게 각 미디어들의 프리젠테이션 시작시간과 종료시간을 요구한다.

나) 스케줄러로부터 구해진 각 미디어들의 시작 시간

을 체크하여 각 미디어 플레이어(media player)들에게 그 미디어 수행시간의 정보와 시작신호(START)를 보내고, 객체의 상태를 running 상태로 변경한다.

다) 각 미디어 플레이어는 미디어를 수행시키기 위해 타이머를 작동시키며, 정해진 시간 만큼 미디어가 수행된 후, 미디어의 수행이 완료(DONE)되었다는 신호를 프리젠테이션 관리자에게 보낸다.

라) 프리젠테이션 관리자는 미디어 플레이어로 부터 완료 신호를 받고, 객체의 상태를 running 상태에서 complete 상태로 변경하고, 미디어 플레이어에게 STOP 신호를 보낸다.

마) 미디어 플레이어는 STOP 신호를 받고, 수행을 종료한다.

바) 프리젠테이션 관리자는 다음에 수행해야 할 미디어를 검사하여 그 미디어 플레이에게 시작 신호와 미디어 수행시간의 정보를 보내 위와 같은 과정을 수행하게 한다.

3) 사용자 상호작용의 처리

프리젠테이션 관리자는 사용자 상호 작용에 의해 프리젠테이션의 순서가 동적으로 변환하는 경우에, 각 미디어 객체의 상태를 관리하고 이를 효과적으로 처리해준다.

이벤트 모니터는 실행시간에 사용자로부터 오는 상호작용을 감시하다가 한 이벤트가 발생되는 경우, 프리젠테이션 관리자에게 신호를 보낸다. 프리젠테이션 관리자는 사용자 상호 작용에 대한 신호를 받고 이에 대한 처리를 시작한다. <그림 9>는 각 이벤트에 대한 처리 루틴을 나타낸다.

Procedure Event_process()

/* i : *a media object,* k : *one of related objects with object i,* E_t : *current time at the point of event occurred,* S_t(i) : *presentation start time of i,* F_t(i) : *presentation finish time of i,* D(i) : *duration of i,* D_r(i) : *remaining presentation time for i,* Sk_t : *time after skip,* Pt_g : *presentation time gap as scaling the speed,* V : *speed of presentation,* △V : *scaling factor of presentation speed,* D_r'(i) : *remaining presentation time for i as the result of scaling event,* FZ : *freeze,* RS : *resume,* F_SK : *forward skip,* B_SK : *backward skip,* SS : *scaling* */

```
begin
  case  event-type : FZ
      for all objects that the current object state is run
          send STOP signal to the media players
          change the object state from run to ready
          compute the duration with remaining time
      D_r(i) = D(i) - |E_t - S_t(i)|
  endcase
  case  event-type : RS
     for all objects that the current object state is ready,
        send START signal and D_r(i) to the media
           players
        change the object state from ready to run
  endcase

  case  event-type : F_SK
     for all objects that the current object state is run,
          send  STOP signal to the media players
          change the object state from run to ready
     compute the skip time
     check the objects to be processed after the skip

  time
     compute the duration with remaining time
     if (S_t(i) < Sk_t) and (F_t(i) > Sk_t) then
          if (S_t(i) < E_t) then
             D_r(i) = D(i) - (|S_t(i) + E_t| + skip time)
          endif
          if (S_t(i) > E_t) then
             D_r(i) = D(i) - (skip time - |S_t(i) - E_t|)
          endif
     endif
     if (D_r(i) < skip time) then
        change the object state from ready to complete
     for the objects to be processed after the skip time,
        send START signal and D_r(i) to the media
players
        change the object state from ready or idle to run

  endcase

  case  event-type : B_SK
     for all objects that the current object state is run,
          send STOP signal to the media players
          change the object state from run to ready
          compute the skip time
          check the objects to be processed after the skip
time
     compute the duration with remaining time
     if (S_t(i) < Sk_t) and (F_t(i) > Sk_t) then
          if (F_t(i) > E_t) then
             D_r(i) = skip time + (D(i) - (|E_t - S_t(i)|))
          endif
          if (F_t(i) < E_t) then
             D_r(i) = D(i) - (|(E_t - skip time)- |S_t(i)|)
          endif
     endif
     for the objects to be processed after the skip time,
        send START signal and D_r(i) to the media
```

```
players
        change the object state from ready or idle to run
    endcase
    case event-type : SS
        for all objects that the current object state is run,
          reset duration
          D_r'(i) = D_r(i) * V * △V
          Pt_g = D_r(i) - D_r'(i)
          D_r'(k) = D_r(k) + Pt_g
    endcase
end
```

<그림 9> 사용자 상호작용에 대한 수행 알고리즘

(가) 일시정지 및 재수행

사용자가 프리젠테이션의 수행을 일시 정지하라는 명령을 입력하면, 프리젠테이션 관리자는 현재 프리젠테이션 중인 모든 미디어를 중단시키기 위해 현재 객체 상태가 run인 모든 미디어 객체에 대한 미디어 플레이어에게 STOP 신호를 보내고, 객체 상태를 run에서 ready로 변경한다. 잔여시간으로 미디어의 수행시간을 고정시킨다.

사용자가 재수행 명령을 입력하면, 현재 일시정지 상태에 있는 모든 객체의 수행을 재개하기 위해 현재 객체 상태가 ready인 모든 미디어 객체에 대한 미디어 플레이어에게 START 신호를 보내고, 객체 상태를 ready에서 run으로 변경한다.

(나) 전진방향 스킵

사용자가 현재 프리젠테이션 중인 미디어의 수행을 전진방향 스킵하라는 명령을 입력하면, 현재 수행중인 모든 미디어 객체들에 대한 미디어 플레이어에게 STOP 신호를 보내 프리젠테이션을 정지시키고 객체 상태를 run에서 ready로 변경한다. 스킵시간을 계산한 후, 스킵시간이 지난 다음에 프리젠테이션 되어야 할 미디어 객체들을 체크하여 각 미디어들의 수행시간을 남은 시간으로 갱신한다. 또한 그 미디어들을 수행시키기 위해 그 미디어 플레이어들에게 START 신호를 보내고, 객체들의 상태 정보를 ready에서 run으로 혹은 idle에서 run으로 변경한다.

(다) 후진방향 스킵

현재 수행중인 모든 미디어 객체들에 대한 미디어 플레이어에게 STOP 신호를 보내 프리젠테이션을 정지시키고 객체 상태를 run에서 ready로 변경한다. 스킵시간을 계산한 후, 스킵시간 이전에 프리젠테이션 되어야 할 미디어 객체들을 체크하여 각 미디어들의 수행시간을 갱신한다. 또한 그 미디어들을 수행시키기 위해 그 미디어 플레이어들에게 START 신호를 보내고, 객체들의 상태 정보를 ready에서 run으로 혹은 idle에서 run으로 변경한다.

(라) 프리젠테이션 속도 조절

사용자가 프리젠테이션 수행 중인 미디어의 프리젠테이션 속도를 늦추거나 빠르게 하는 프리젠테이션 속도 조절 명령을 입력하면, 지정된 시간 범위에서 수행되어야 할 모든 미디어들을 체크하여 그 미디어의 수행 속도를 변화시킨다. 또한, 지정된 시간 뒤에 수행될 예정인 미디어의 속도는 변경되지 않지만 동기화를 위해 프리젠테이션 속도의 조절비에 따라 프리젠테이션 시간을 연장 또는 단축시켜야 하는 등의 프리젠테이션 변화가 반영된다.

5. 결론 및 향후 연구 방향

본 연구에서 설계 및 구현한 시스템은 인과성 시간 관계를 사용하여 실시간으로 객체의 수행시간 변경이 가능함으로써 융통성 있는 프리젠테이션 실행을 제공했다. 또한 아이콘 형태의 시간 관계 명세를 통해서 일반 사용자도 쉽게 저작하고 수정할 수 있는 사용자 인터페이스를 제공하고 있다. 또한 저작중에 발생될 수 있는 오류나 기타 시간의 불일치성 등을 자동으로 검증하여 즉시 사용자에게 알려주는 동적 저작 작업 환경을 제공하였다.

멀티미디어 문서를 편집하고 효과적으로 관리하기 위해서 작성된 문서를 그 의미에 맞게 내부 구조화하였다. 이러한 문서 파싱 작업을 위하여 본 연구에서는 트리 구조의 계층적인 형태를 이용하여 문서를 저장하고 관리하였다. 파싱 단계에서 생성된 트리 구조를 가지고 프리젠테이션 엔진에서는 실행시간에 동기와 비동기를 고려하여 스케쥴하고 프리젠테이션을 수행하도록 한다. 특히 프리젠테이션 엔진의 스케쥴러는 각 미디어들의 프리젠테이션 시작 시간을 스레드 이진 트리를 이용하여 효과적으로 구하고 있다.

시스템 구현은 마이크로소프트사의 윈도우즈'95 환경하에서 프로그램 개발 도구로 비주얼 C++ 4.2 을 사용했다. 이미지나 텍스트 미디어 처리는 MFC (Microsoft Foundation Class) 4.2 라이브러리를 활용하고, 비디오와 오디오 미디어 처리를 위한 .avi 파일, .wav 파일은 win32.Api의 미디어 제어 인터페이스(MCI) 함수를 사용했다.

참 고 문 헌

[1] J. F. Koegel and J. M. Heines, "Improving Visual Programming Languages for Multimedia Authoring," EDMEDIA'93, 1993, pp. 286-293.

[2] R. Steinmetz and K. Nahrstedt, Multimedia: Computing, Communications, and Applications, Prentice Hall, 1995, Chap. 13-18.

[3] John F. Koegel Buford, Multimedia Systems, ACM Press, Addison-Wesley, New York, 1994, Chap. 7, 11, 12.

[4] Multimedia Authoring System FAQ, http://www.tiac.net/users/jasiglar/MMASFAQ .html.

[5] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," CACM, Vol. 26, No. 11, 1983, pp. 832-843.

[6] Causal-Relation, http://www.darmstadt. gmd. de/publish/komet/gen-um/ node 148-152.html.

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

[8] G. Blakowski, J. Hubel, U. Langrehr, and M. Muhlhauser, "Tool Support for the Synchronization and Presentation of Distributed Multimedia," Computer Comm.., Vol. 15, No. 10, Dec., 1992, pp. 611-618.

[9] Thomas D. C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," IEEE Journal on Selected Areas in Comm., Vol. 3, No. 8, Apr., 1990, pp. 413-427.

[10] Horowitz, Ellis, et al, Fundamentals of data structures in C, Computer Science Press, 1993.

[11] 최숙영 외 1인, "인과성 관계에 기반한 멀티미디어 프리젠테이션 시스템," 한국산업정보학회 논문지, 제3권 2호, pp.111-119

[12] J. Schnepf, J. Konstan, and D. Du, "Doing FLIPS: Flexible Interactive Presentation Synchronization," IEEE Journal on Selected Areas in Comm. Jan., 1996, pp. 114-125.

[13] L. Weitzman and K. Wittenburg, "Automatic Presentation of Multimedia Documents using Relational Grammars," Multimedia'94, 1994, pp. 443-451.

[14] Boring A., et al, "Constraint-Based Tools for Building User Interfaces," ACM Trans. on Graphics, Vol. 5, No. 4, Oct., 1986, pp. 345-374.

최 숙 영

1988년 8월 전북대학교 이학사 (전산학)
1991년 2월 전북대학교 대학원 이학석사(전산학)
1996년 2월 충남대학교 대학원 이학박사(전산학)
1996년 ~ 현재 우석대학교 컴퓨터교육과 조교수
관심 분야 : 멀티미디어응용, 원격교육, 병렬처리

신 현 산

1984년 2월 충남대학교 계산통계학과 계산학 학사
1986년 2월 충남대학교 대학원 계산통계학과 계산학 석사
1998년 8월 충남대학교 대학원 전산학과 전산학 박사
1988년 ~ 1995년 한국전자통신연구원 근무
1995년 ~ 현재 충청대학 컴퓨터학부 조교수
1999년 ~ 현재 충청대학 사이버캠퍼스 추진본부장
관심 분야 : 멀티미디어 저작시스템, 가상대학, 병렬처리, 부호처리

유 관 종

1976년 서울대학교 계산통계학과 (이학사)
1978년 서울대학교 계산통계학과 (이학석사)
1985년 서울대학교 계산통계학과 (이학박사)
1989년 ~ 1990년 캘리포니아 대학교(Irvine) 방문교수
1979년 ~ 현재 충남대학교 컴퓨터 과학과 교수
관심 분야 : 멀티미디어 응용, 병렬처리, 에이전트 시스템