멀티미디어를 기반으로 하는 저작도구 툴북에서 객체 자동 변환을 이용한 자동 프리젠테이션 시스템 개발

양 옥 렬[†] · 정 영 식[††] · 이 용 주[†††]

요 약

멀티미디어 저작도구를 이용한 응용 프로그램 개발에 있어서, 기존의 저작도구들 중에 멀티미디어 툴북의 경우 객체들이 사건을 기반으로 하는 상태 변환 방식을 사용하므로 인해 프리젠테이션을 자동으로 가능하게 할 수 있는 기능을 제공하지 못한다. 본 연구에서는 객체의 자동 변환 기법을 통하여 자동 프리젠테이션 기능을 멀티미디어 저작도구가 시스템 DLL(Dynamic Link Library) 형태로 제공하도록 하기 위한 객체 자동 변환 기법을 개발하고자 이를 멀티미디어 툴북의 페이지 전환 및 미디어 제어 인터페이스를 이용하여 오디오 자원의 상태 변환 기능을 추가하여 멀티미디어 저작도구의 기능 확장이 가능하게 하였다.

Development of Auto Presentation System of Toolbook Using Object Auto Transition on Multimedia Authoring Tool

Ok-Yul Yang[†] · Young-Sik Jeong[††] · Yong-Ju Lee[†††]

ABSTRACT

When we present some information, we can use application programs through multimedia-based authoring tools. Especially, many programers proposed to improve its integration time and reduce programming speed and easy to use. However, multimedia based authoring tools have not all of programming methodologies and do not supply special functions from user's request. Therefore, we have to apply effective functions through high-level programming languages.

In this paper, we propose to use small application programs through dynamic linking methods. So we reduce overhead from memory loading. In authoring tools, we can use MCI(media control interface) call functions for playback audio files. We development ATS(Auto Transition System) for several functions-close MCI call audio files, get object status, page-to-page transition. We evidently show that an optimal configuration of presentation obtained by ATS algorithm.

1. 서 론

응용 프로그램의 개발에 있어서 빠르게 변화하는 주변의 시스템이나 사용자 환경에 만족하기 위한 개발 도구로서 멀티미디어 저작도구(authoring tool)를 이용한 응용 프로그램 개발이 활발해지고 있다[4, 5, 13, 14]. 멀티미디어 저작도구의 가장 큰 특징은 전문적인 개발능력 없이도 응용 프로그램의 개발 시간을

† 준 회 원:원광대학교 컴퓨터공학과 박사과정
†† 종신회원:원광대학교 컴퓨터공학과 교수
††† 정 회 원:원광대학교 컴퓨터공학과 교수
논문접수:1996년 12월 2일, 심사완료:1997년 2월 19일

단축시키는 것이다[1, 2]. 최근 들어 저작도구를 이용하여 가장 적은 시간에 가장 효율적으로 학교나 회사 기타 홍보나 프리핑을 위해 필요한 프리젠테이션 형태의 응용 프로그램 개발 중 특히 프리젠테이션의 효과를 높이기 위해 별도의 프리젠테이션 설명자 없이 자동으로 프리젠테이션 하기 위한 방법이 필요하게 되었다. 그러나 기존의 저작도구들이 갖는 기능으로는 객체들의 프리젠테이션을 자동으로 변환하게 하는 기능을 제공하는 데에는 여러가지 어려운 점이 있다. 본 연구는 객체 자동 변환 기법을 이용하여 자동 프리젠테이션 기능을 멀티미디어 저작도구에 동적 연결 라이브러리를 이용한 객체 자동 변환 시스템을 개발하여 응용 프로그램에서 발생하는 오버헤드(overhead)를 줄이고 효율적인 시스템 관리가 가능하도록 하는데 그 목적이 있다.

저작도구 안에서 사용되는 데이타는 텍스트, 그래픽, 애니메이션, 오디오 및 동영상 등의 단일 미디어에 대한 기능적 표현보다는 각각의 모노미디어(monomedia)가 결합된 형태인 멀티미디어 데이타는 데이타의 포맷이 디지탈 형태로 만들어져 있으며, 인터택티브(interactive) 기능과 대량의 데이타를 보관, 전송, 표현 기능, 실시간처리(real-time processing) 등을 기반으로 하여 데이타에 대한 통합성, 공존성, 독립성, 상호작용성(interactivity)의 특징을 갖는다[6]. 미디어간의 동기화는 하나의 미디어가 다른 미디어의 동작이나 상태 변경에 영향을 미치는 미디어간 관계(intermedia relationship), 동영상이나 오디오 같이 복수개의 미디어가 병렬적으로 표현되는 미디어 협력(media cooperation), 그리고 하나의 미디어가 또다른 미디어로 전환되는 미디어 변환(media conversion) 등의 세 가지 경우에 적용된다[5]. 또한 멀티미디어 동기화의 특성에 따라 미디어는 시간종속적 미디어(time-based media)와 텍스트와 오디오의 결합과 같은 공간종속적 미디어(spatial-based media)로 구분된다[6]. 따라서 본 연구에서는 이러한 멀티미디어 특성에 적합하며 세계적으로 가장 많은 사용자와 개발 응용 프로그램을 갖고 있는 Asymetrix사의 멀티미디어 툴북(Multimedia ToolBook) 4.0 CBT Edition에서 자동 프리젠테이션을 위한 객체의 자동 변환 기법을 고안하고 이를 시스템 DLL(dynamic link library) 형태로 생성하였다.

2. 멀티미디어 저작도구의 분류 및 기능

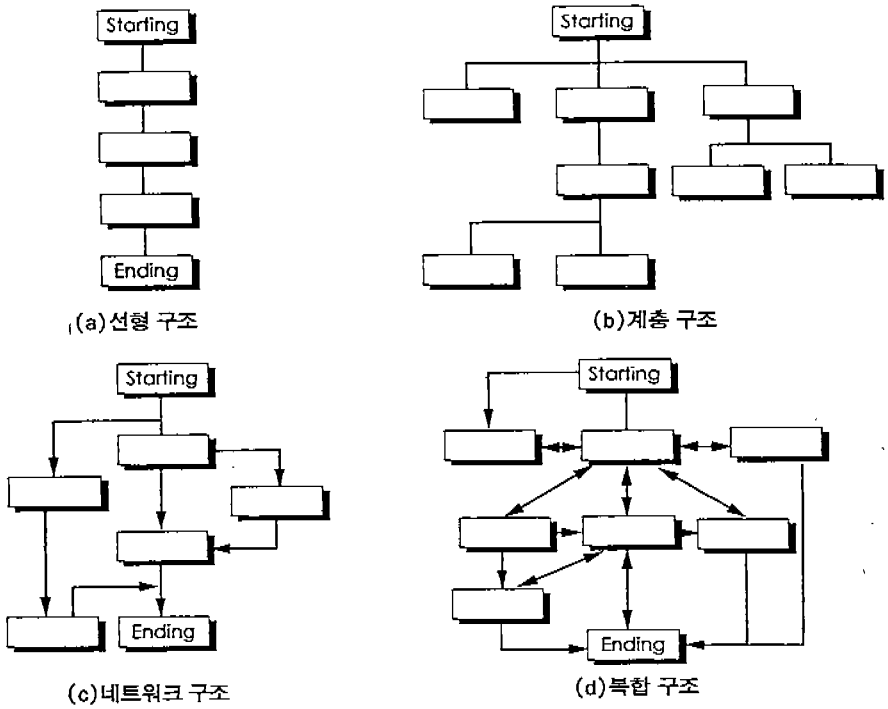
2.1 저작도구의 분류

일반적으로 사용자는 멀티미디어 저작도구가 가지고 있는 메타포어(metaphor) 특징을 조사하고 개발하려고 하는 응용 프로그램에 적합한 개발도구를 결정한다. 메타포어는 저작도구가 갖게되는 전체적인 저작 형식을 조사한 후 특징별로 은유(metaphor)화 하여 구분해 놓은 분류인데, 여기에는 흐름도(flowchart metaphor), 책(book & page metaphor), 시간선(timeline metaphor) 방식 등으로 구분된다[13]. 또한 멀티미디어 저작도구는 저작 기반 도구 사용에 따라 페이지 혹은 스크립트 기반 저작도구, 아이콘 기반 저작도구, 시간기반 저작도구로 구분하거나 여기에 카드 기반 저작도구를 추가하기도 한다. 또한, 프로그래밍 방법에 따라 (그림 1)과 같은 선형구조, 계층구조, 네트워크 구조 및 복합 구조 등 네 가지 형태로 세분화된다[5, 13].

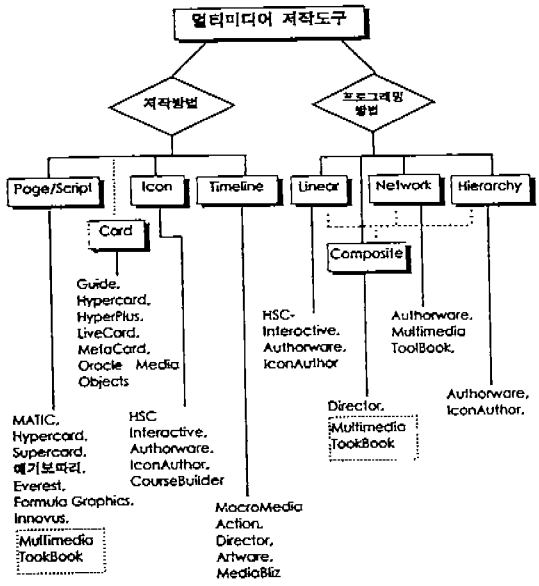
멀티미디어 저작도구에 대한 분류는 다음 (그림 2)와 같다. 일반적으로 멀티미디어 저작도구들은 텍스트, 다큐멘트, 사운드, 그래픽, 애니메이션 및 동영상 처리를 위한 도구를 포함하고 있으나 이들 모두를 갖추고 있는 저작도구는 없으므로 개발하고자 하는 응용 프로그램에서 가장 많은 사용을 요구하는 미디어 처리 도구를 제공하는 저작도구를 이용한다.

2.2 저작도구의 기능 및 특성

일반적으로 멀티미디어 저작도구의 기능에는 미디어 자체(intramedia)나 미디어 장치간(intermedia) 연결기능, 사용자 입력에 관한 미디어 제어기능, 미디어 파일의 동시 실행 기능(concurrent playback) 등이 있다[1, 2]. 여기서 가장 중요한 기능은 각 모노 미디어를 결합하여 동기화(synchronization) 시키는 동시 실행 표현 기능이다[14]. 이러한 동기화가 어긋나게 되면 사람이 말하는 입술 모양이나 동작이 소리와 서로 어긋나는 어색한 상황이 발생하게 된다. 즉 둘 이상의 미디어를 사용하여 이를 동시에 재생시키면 이러한 재생은 하나의 플랫폼(platform)에서 이루어진다. 또한 멀티미디어 데이타와 이를 재생(playback)시키는 미디어 제어(media control)의 순조로운 결합이 필요하고 저작도구로 만들어진 응용프로그램을 사용하



(그림 1) 저작방법에 따른 프로그래밍 구조 다이어그램
(Fig. 1) Diagram of programming structure on authoring methods



(그림 2) 저작도구의 분류
(Fig. 2) Hierarchy of authoring tools

는 임의의 사용자로 부터 받게 되는 입력상황에 대한 제어 기능을 제공해야 한다. 따라서 저작도구가 갖는 미디어에 관한 기본적인 4가지 기능을 통합한 멀티미디어의 다양한 기능이 요구 된다.

멀티미디어 저작도구를 사용하는 개발 시간을 일반 컴파일러 프로그래밍 언어에 비해 인터프리터 방식을 채용함으로써 응용 프로그램의 개발 시간을 1/3 정도로 단축시킬 수 있으며 프로그램의 작성 후에도 버그가 작다. 또한 객체별로 프로그래밍을 할 수 있으므로 구조적 프로그램이 가능하며 따라서 인원의 효율적 분배를 통한 오디오, 텍스트, 동영상, 애니메이션 등의 계한된 미디어 분야별로 다루어 질 수 있다는 점이다. 또한 이러한 인터프리터 언어가 갖는 특정인 사용자와 시스템간의 대화성이 강화되어 있다.

2.3 문제점

앞서 언급한 (그림 2)의 구조화된 저작도구(structured authoring tool)의 분류에서 멀티미디어 응용 프로그램 개발을 위해 사용되는 시간선(timeline), 아이콘(icon) 방식 등의 저작방법을 사용하는 경우에 이에 대한 프로그래밍 방법으로 선형(linear), 계층(hierarchy) 또는 네트워크(network) 형식의 프로그래밍 방법을 이용한다. 이러한 저작 방법의 경우에 응용 프로그램의 진행이 시간이나 계층적 트리 구조를 따라 상태나 환경의 변화가 아무런 이벤트 없이 자동으로 가능해지므로 특수한 프로젠테이션 자동화 기법의 기능 추가 없이도 자동 프리젠테이션이 가능하다. 그러나 페이지(page)나 스크립트(script) 혹은 카드(card)와 같은 저작 방법을 이용하는 경우 프로그래밍 구조가 객체 지향적 성향을 띈다. 또한 이러한 페이지 및 카드 형태를 갖는 저작 방법의 경우 (그림 1)의 복합 구조(composite) 형태의 프로그래밍 방법과 같이 상태 변화가 많은 응용 프로그램의 경우 시간선 방식이나 아이콘 방식을 사용하는 경우보다 융통성을 갖는다. 그러나 이들은 각각의 페이지나 카드 하나하나가 객체에 해당하므로 각 객체를 실행하기 위한 각각의 사건, 즉 이벤트가 필요하다.

페이지를 객체로 하는 멀티미디어 툴북은 주로 교육용(CAI : Computer Assisted Instruction) 타이틀, 종합 안내 시스템인 키오스크(KIOSK), 게임, 데이타베이스 전위 시스템, 오락(entertainment), 그리고 교육과 오락을 결합한 에듀테인먼트(edutainment) 등의 응용 프로그램 개발이 용이하다. 그러나 인터프리터 언어는 일반 컴파일러 언어를 사용한 응용 프로그램에 비해 수행속도가 저하되며, 저작도구 자체가 갖는 일반적으로 특정적인 기능 중심으로 설계되어 응용 프로그램 개발을 위한 융통성이 적고 이에 따라 객체 제어 능력이 미약하며, 필요한 기능을 완벽하게 구현하거나 제어 하기가 불가능하다. 또한 일반적으로 사건이라는 이벤트에 의해 상태를 변환하므로 사건이 발생하지 않는 상황에서의 상태 변환의 어려움이 있다. 따라서 자동 프리젠테이션 기능을 제공하는데 어려움이 있으며 멀티미디어 저작도구의 기능으로 연동 시키는 것이 필요하다.

3. 객체 변환 메카니즘 및 자동 프리젠테이션

3.1 객체 변환방법

멀티미디어 툴북에서의 객체(object)는 크게 graphic, field, button, page, background의 5가지 객체로 구성된다. 툴북은 사건-중심(event-driven)방식으로 프로그램을 실행시키며 이 5가지 객체는 응용프로그램의 진행 상태에 따라 혹은 이벤트 발생 여부에 따라 프로그램에 반응한다. 즉, 키보드를 누르거나 마우스를 클릭하거나 하는 이벤트를 의미하며 상황에 따라 휴지상태(idle state)에 있을 때도 특정 이벤트를 발생시키고 이에 따라 파일 안에 저장된 미디어들이나 객체가 그 상태를 변환하거나 페이지로의 이동이 가능하게 한다. 이러한 객체의 변환 과정 중 특히 페이지 대 페이지의 이동은 툴북이 페이지 단위의 객체지향적 스크립트 언어이므로 페이지를 관리하고 운영하는 핸들러(handler)의 핸들링이 중요하다. 만일 하나의 페이지에서 열어놓은 클립이나 객체가 제대로 닫혀지지 않고 다른 페이지로 이동되면 핸들러의 중복으로 프로그램이 제대로 수행되지 않게 된다. 따라서 단 한번의 사건에 따른 목적 페이지까지 자동 페이지 변환이 필요하며, 이를 멀티미디어 저작도구의 주요 미디어 기능인 연결, 제어, 동시실행, 동기화 등 4가지 기능에 부가적으로 기능을 추가하는데 그 목적이 있다(그림 3).

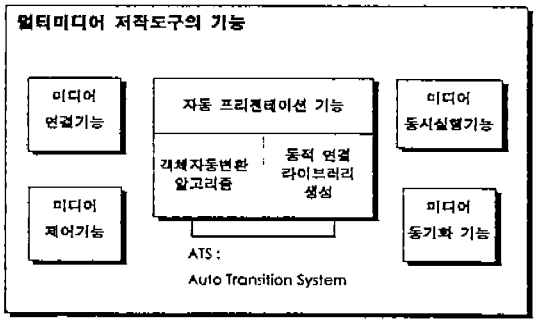
툴북에서는 MCI(media control interface)를 이용한 핸들러의 자원 호출 사용이 가능하며 이는 다음과 같

은 callMCI()함수를 이용하여 애니메이션(animation), CD-DA, 디지탈 비디오, 미디 시퀀서, 비디오 오버레이, 비디오 디스크, 웨이브폼 오디오 포멧 등의 7가지 MCI 호출이 가능하다.

■MCI

callMCI("⟨MCI command⟩⟨Device⟩⟨Command argument⟩
[wait]"
[, ⟨notify object reference⟩])

MCI 호출에는 다음과 같은 MCI 명령어가 있으며 특정 자원(Resource)의 상태는 호출된 MCI 명령어에 의해 좌우된다⟨표 1⟩.

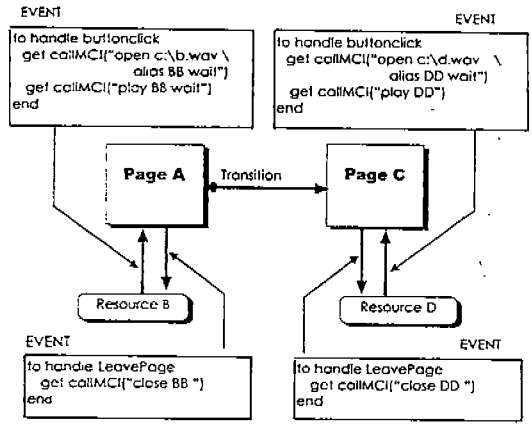


(그림 3) 멀티미디어 저작도구의 기능 확장
(Fig. 3) Extention of functions on multimedia authoring tool

⟨표 1⟩ 일반적인 MCI 명령어
⟨Table 1⟩ MCI general commands

Command	Mean	Command	Mean
Open	Device initialization	Seek	Move to special file location
Close	Device close	Set	Option setting
Break	Passing control driver to application	Sound	Playback sound
Capability	Device capability	Status	Device status information
Info	Device information	Stop	Device playback stop
Pause	Device playback pause	Sysinfo	Get MCI system info.
Play	Device playback		

필요에 따라 여러 개의 자원이 하나의 페이지나 응용 프로그램에서 사용될 수 있는데 (그림 4)과 같이 A 페이지에서 MCI로 호출된 B라는 오디오 자원이 재생되고 있는 상태에서 C라는 페이지로 이동하면서 MCI로 D라는 오디오 자원을 호출한다면 이때 B오디오 자원은 A라는 페이지를 떠나면서 닫아야 한다. 따라서 어떠한 형태의 MCI에 의한 자원 호출이 발생해도 이를 자동으로 페이지 변환에 따라 열기(open), 닫기(close)를 할 수 있는 새로운 MCI 호출에 관한 메카니즘이 필요하다.



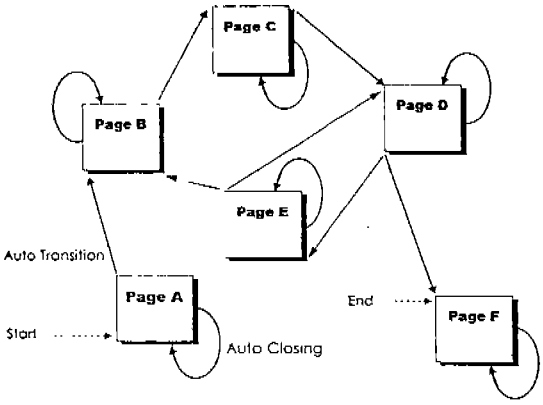
(그림 4) 페이지 변환 다이어그램 및 사용예
(Fig. 4) Diagram of page transition and examples

3.2 객체 자동 변환 알고리즘

객체 자동 변환 알고리즘이란 한번의 외부 입력 사건 발생(start event)을 기점으로 아무런 외부 입력 없이 객체 자동 변환 및 MCI 호출 자원의 자동 닫기가 가능하게 하는 알고리즘을 의미한다. 객체 자동 변환에 대한 개념적 구성은 다음 (그림 5)와 같고 그에 대한 알고리즘은 [알고리즘 1]과 같다. 복합구조 형태로 프로그래밍 된 응용 프로그램은 미리 설정 되어진 프로그램에 따라 페이지의 항해(navigation) 순서를 지정하고 페이지의 변환은 단순한 순차적인 변환 뿐 아니라 메인 페이지에서 부 페이지로의 페이지 변환 후에 다시 메인 페이지로 변환하고 이는 다시 또다른 부 페이지로의 페이지 자동 변환이 발생할 수 있게 할 수 있다.

여기서 이용되는 MCI 호출은 특별히 MCI 자원 중에 오디오(sound) 자원 부분에 한하여 고려한 알고리즘으로 웨이브 폼으로 만들어져 있는 파일에 대한 자동 열기와 닫기의 기능이 가능하도록 하기위한 알고

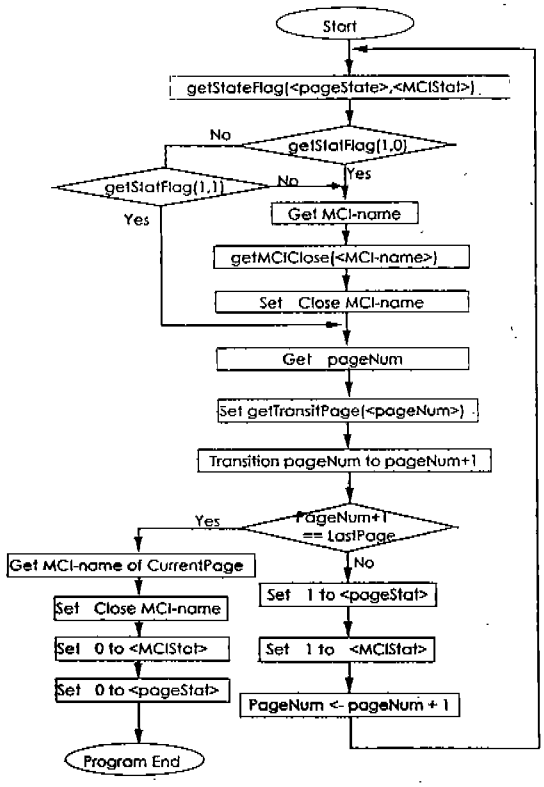
리즘이다. 따라서 MCI를 통해 호출되어지는 자원을 갖고 있는 툴북의 페이지에서의 자원 상태 분석과 상태 해석을 통해 해당 페이지에서의 부분적인 자동 프리젠테이션이 가능해 진다. 자동 프리젠테이션을 위한 객체 자동 변환기에 대한 [알고리즘 1]을 단계별 자동 프리젠테이션 시스템(ATS: Auto Transition System)을 흐름도 방식으로 구성하면 (그림 6)과 같다.



(그림 5) 자동 객체(페이지) 변환 다이어그램
(Fig. 5) Diagram of auto object(page) transition

[알고리즘 1] 객체 자동 변환 알고리즘
[Algorithm 1] Object auto transition algorithm

```
[알고리즘 1.] 객체 자동 변환 알고리즘

Step 1:  페이지의 상태 플래그 값을 얻는다.
         Get getStateFlag (<pageStat>, <MCIStat>)
            if getStateFlag( 1, 0) then
               goto Step 2
            else-if getPageFlag(1,1) then
               goto Step 4

Step 2:  MCI 호출 자원 이름을 추출한다.
         Get MCI-name
         Set getMCIClose(<MCI-name>)
         Set Close MCI-name

Step 3:  현재 페이지 이름을 추출한다.
         Get pageNum
         Set getTransitPage(<pageNum>)
         Transition pageNum to pageNum+1
            if pageNum+1 is Lastpage then
               goto Step 6
            else
               goto Step 4

Step 4:  Set  0  to  <pageStat>
         Set  1  to  <MCIStat>
         pageNum <- pageNum + 1

Step 5:  Loop Step 1

Step 6:  Get  MCI-name of CurrentPage
         Set  Close MCI-name
         Set  0  to  <pageStat>
         Set  0  to  <MCIStat>
         Program End
```



(그림 6) 객체 자동 변환 알고리즘의 단계별 흐름도
(Fig. 6) Step-by-step flowchart of ATS algorithm

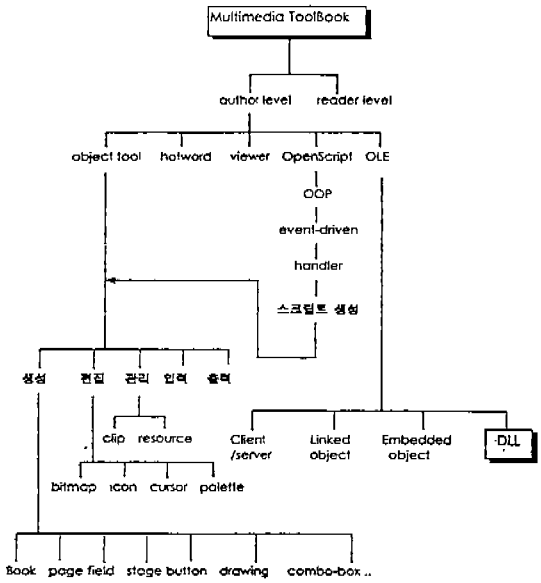
현재 툴북의 변환 기법에는 단순히 페이지에서 페이지의 전환을 수행하는 "Transition" 명령이 있다. 이는 페이지 이동에 단조로움을 피하기 위한 간단한 화면의 변화 효과로 원하는 선택 페이지로의 전환만이 가능하다. 이는 이전 페이지에서 다음 페이지로의 이동에 있어서 전혀 이전 페이지의 상태를 고려하지 않는 변환 기법으로 페이지라는 객체는 바뀌지만 실제 페이지에서 수행되는 MCI 호출 자원의 상태나 그 밖에 클립들과의 인터페이스도 갖지 않으므로 이 방법만으로는 자동으로 프리젠테이션 되는 응용 프로

그램에 적절한 기능을 제공할 수 없다. 이렇게 페이지 마다 발생하는 사건에 대하여 개별적 객체별로 스크립트를 써주어야 하는데 이렇게 객체의 이름만 다르고 모든 스크립트의 내용이 동일할 때 이를 위한 함수 형태의 스크립트를 필요로 한다. 따라서 본 연구에서는 툴북에서 이루어지는 페이지 단위의 객체 변환에서 하나의 페이지가 포함하는 graphic, field, 혹은 group 등의 객체가 하이퍼텍스트(hypertext)나 하이퍼미디어(hypermedia), 그리고 핫 워드(hot word) 등의 다양한 객체 상황에서의 MCI 호출을 통한 자원의 상태를 열기, 동작(실행), 닫기 등이 자유롭게 수행하도록 하는 시스템 라이브러리 형태인 DLL(dynamic link library)의 형태로 구현하여 시스템의 전역 환경 속으로 포함시켜 페이지의 변환이 생기면 자동으로 이전 페이지의 MCI 호출 자원을 닫아 주는 역할을 할 수 있도록 하는 기능을 추가한다.

3.3 DLL 생성 방법

3.3.1 툴북 환경에서의 DLL

툴북에서는 OLE(object linking and embedding)를 이용한 객체 사용이 가능한데 이러한 OLE의 사용은 만들어진 객체의 특정 부분에 대한 편집이나 수정이 그 객체가 만들어진 처음 상태에서 가능하도록 해준다. 이는 예를 들어 툴북에서 페인트 객체(paint object)로 그려진 그림의 수정이 마우스의 더블클릭에 의해 자동으로 paint object editing program이 실행되면서 paint 객체의 수정 및 편집이 가능해 진다. 이렇게 툴북이 가지고 있는 기존의 기능에서 OLE에 대한 객체 응용 기능에는 client/server, linked object, embedded object, DLL 등의 네 가지 방법이 있으며 이는 (그림 7)와 같다. client/server는 Microsoft Excel이나 Word, Paintshop 등에서 제공(server)하는 객체의 사용(client)만이 가능하며, linked object는 server application에서 사용하는 자료의 파일을 툴북에서 참조하여 사용하는 경우에 사용할 수 있다. 또한 embedded object는 server application에서 만든 자료의 복사본을 툴북 파일에 포함시켜 사용하는 것이 linked object와 다른 점이다. 마지막으로 DLL은 툴북을 이용한 응용 프로그램에서 새로운 기능을 생성하고 이를 툴북에서 사용하고자 할 때에는 동적 연결 라이브러리인 DLL을 이용하도록 하고있다.



(그림 7) 멀티미디어 툴북의 객체 단위 세부 기능 분류
(Fig. 7) Hierarchy of substantial object functions in multimedia toolbook

3.3.2 새로운 알고리즘을 이용한 DLL

툴북에서는 윈도우즈 환경에서의 동적 환경을 통해 메모리에 단일 프로세스만을 적재하여 메모리에 중복된 프로세스를 없애고 프로그램을 적재하는데 걸리는 시간(memory loading time)과 메모리 낭비를 최소화 하기 위해 DLL을 사용한다. 또한 자주 사용되지 않는 기능을 디스크에 저장해 두었다가 필요할 때마다 불러쓰기 위해 이를 사용한다. 현재 제공되는 DLL은 TB30DLG.DLL, TB30DOS.DLL, TB30WIN.DLL 등이 있다. 이들 DLL을 통해 다이얼로그 박스를 생성하고 제어하거나 도스상의 파일이나 디렉토리 관리, 윈도우 실행 환경을 설정하고 디스플레이 시키거나 글꼴의 세부 선택 사항 지정 등의 기능을 제공하고 있다.

본 연구에서 구현하려는 기능을 위해 [알고리즘 1]에서 제시된 알고리즘을 이용하여 페이지 자동 변환과 MCI 호출 자원의 자동 closing 기능을 수행하는 기능을 "Transit.dll" 이름으로 시스템 DLL화 시킨다. 멀티미디어 툴북에서는 "Transit.dll"과 같은 동적 연결 라이브러리를 생성하면 필요한 기능을 시스템 중에 EnterBook이나 EnterApplication과 같은 메시지 핸들

러에 설치하여 DLL을 로드하는 부분을 삽입하고 응용 프로그램의 수행이 완전히 끝날 때 LeaveBook이나 LeaveApplication 등의 메시지 핸들러 안에서 이를 제거시키는 방법을 이용한다. "Transit.dll" 사용을 위한 선언은 다음 (그림 8)과 같다.

이와 같이 객체 자동 변환 알고리즘에 의해 만들어지는 동적 연결 라이브러리는 객체 자동 변환기인 ATS의 핵심이 된다. 이렇게 동적 연결 라이브러리를 통해 응용 프로그램의 크기를 최적화 시키며 메모리 적재의 최소화, MCI 호출 자원에 대한 자원 열기/닫기 및 동시성 제어와 같은 상태 제어가 가능해 진다. 또한 이는 페이지 변환의 기능까지 제공한다.

```
To handle EnterApplication
    :linkDLL "transit.dll"
      INT getStateFlag(<INT pageStat>,<INT MCIStat>)
      STRING getTransitPage(<INT pageNum>)
      STRING getM... Close(<STRING MCI-name>)
    :end linkDLL
end EnterAppli...
```

(그림 8) 자동 페이지 변환을 위한 Transit.dll 선언
(Fig. 8) Declare of "Transit.dll" for auto page transition

3.4 자동 프리젠테이션 기법

멀티미디어 툴북에서 자동 프리젠테이션을 위해서는 수십 혹은 수백 개의 페이지로 구성된 매 페이지마다 스크립트를 개별적으로 써주는 것은 매우 힘든 일이다. 따라서 멀티미디어 툴북에서는 자동 프리젠테이션을 위해서 3.2절의 자동 객체 변환 알고리즘을 이용한다. 자동 객체 변환 알고리즘을 이용한 자동 프리젠테이션에 대한 전체적 개념도는 다음 (그림 9)와 같다.

윈도우즈 환경에서 수행되는 저작도구는 먼저 윈도우즈 시스템 환경에 영향을 받으며 멀티미디어 툴북은 실제로 버튼 클릭과 같은 하나의 사건(event)에 대한 메시지(message)를 객체로서 인식하고 이러한 객체가 가질 수 있는 그룹(group)을 찾고, 그룹이 속해있는 페이지(page)와 배경(background)을 통해 멀티미디어 툴북이 갖게 되는 하나의 책(book)이 만들어 진다. 이러한 책은 실제 동작에 필요한 모든 작업을 핸들러의 핸들링에 의해 수행되어 지는데 이러한 핸들러는 OpenScript를 통해 제어한다. 모든 핸들러는 상위의 응용 핸들러(application handler)에 영향을 받게 된다.

이렇게 각 모듈별 스크립트 체제인 모듈 핸들러에서는 페이지별로 작성된 스크립트에서 나타나는 페이지 상태를 알려주는 동적 연결 라이브러리인 "Transit.dll" 호출을 통해 getStateFlag() 함수에서 얻어지는 페이지 상태값 〈pageStat〉와 〈MCIStat〉를 얻는다. 이에 따라 〈MCIStat〉가 set되면 호출된 MCI 자원을 닫고 새로운 페이지로 이동되기 위해 getTransitPage() 함수를 호출한다. 호출된 이 함수는 페이지가 마지막 페이지인가를 확인하고 그렇지 않으면 다음 페이지로 페이지 변환을 수행한다.

동적 연결 라이브러리를 이용하여 툴북을 통해 만들어진 응용 프로그램을 자동으로 객체의 상태를 변환시켜 자동으로 프리젠테이션 해주는 기능을 제공하기 위해서는 시스템 북으로부터 얻은 메시지를 응용 프로그램의 책에서 전달받아 이를 각 페이지별로 MCI 호출 자원의 상태 플래그를 얻어 그 값과 페이지 전환 상태 플래그의 값이 set되어있을 때 이를 자동으로 초기 페이지에서 마지막 페이지에 이르는 진행을 자동으로 연결해 주게 한다. 이러한 페이지 변환에는 각 페이지별 스크립트 핸들러와 해당 책 스크립트 모듈 핸들러의 영향을 받게 된다.

4. ATS(Auto Transition System)의 개발

4.1 ATS의 설계

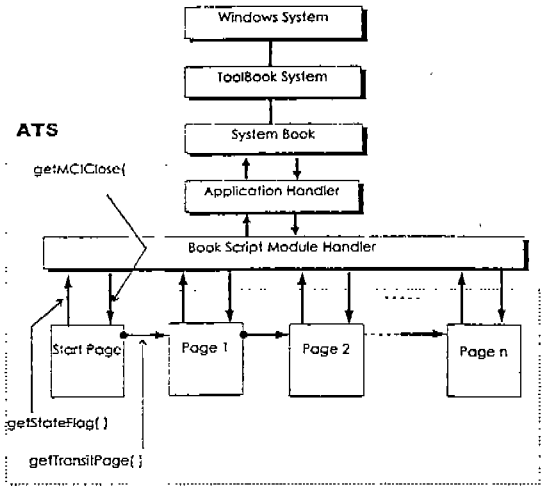
동적 연결 라이브러리를 통하여 툴북에서 구현하고자 하는 객체 자동 변환기(ATS: Auto Transition System)란 입력된 이벤트를 각 페이지 단위별 객체를 모듈 형태로 구분하고 이를 상태 플래그를 통해 자동으로 변환시키는 기능을 의미한다. 본 연구에서 제안하는 ATS은 세부적으로 하위 모듈로 구성되어 있는데 이들 각 모듈은 Using DLL module, Object state analysis module, MCI call module, MCI resource state module, MCI resource play module, MCI resource close module, Page & oblects inf a page state module, Page transition module 등 8개의 세부 모듈이다. 이들의 각 기능을 살펴보면 다음과 같다.

DLL 사용 모듈(using DLL module)은 자동 프리젠

테이션 형태의 응용 프로그램에서 맨 처음 시작을 위해 필요한 키보드의 입력이나 마우스의 클릭 등이 필요하다. 이러한 입력을 이벤트 혹은 사건(event)이라고 할 수 있는데 입력 사건이 실제 사용자 인터페이스에 환경에서 입력을 처리하여 이를 ATS의 세부 모듈로 옮겨 지도록 한다. 이때 처음으로 입력 사건을 처리하는 부분이 바로 이 모듈이며 실제 사용되어질 DLL 파일을 찾아 사용을 정의한다. 여기에서 처음으로 본 연구에서 개발한 "Transit.dll"을 선언하고 함수 getStateFlag()의 상태 값을 추출하게 된다.

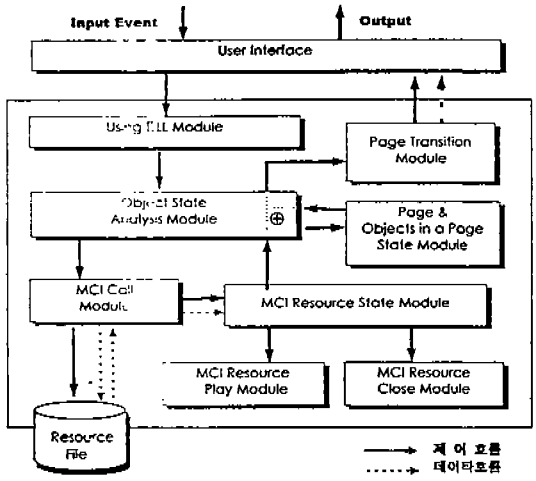
객체 상태 분석 모듈(object state analysis module)은 사용되어지는 동적 연결 라이브러리가 객체의 상태가 어떻게 변화하는가를 체크하고 이에 따른 후속 조치를 수행하는 객체 상태 분석 모듈이다. 이는 툴북에서 사용되는 모든 페이지에서 호출되는 MCI 호출 자원을 찾아 메모리에 적재 시킨다. 또한 메모리에 적재된 MCI 호출 자원의 상태에 따라 페이지 변환 모듈로 상태를 천이 시킨다. MCI 호출 모듈(MCI call module)은 객체 상태 분석 모듈에서 메모리에 적재 시킬 MCI 호출 자원을 실제로 메모리에 적재하는 일을 담당한다. 이 모듈에서 메모리로 자원을 적재하기 위해서는 보조 메모리인 하드디스크나 CD-ROM에 들어있는 파일의 파일 경로(path)를 정확히 찾아내야 한다. 툴북에서는 보조 메모리에서 자원을 접근(access)해 올 때 하드디스크와 CD-ROM의 양쪽에 자원을 갖고 있을 때에는 자원 접근 속도가 보다 빠른 하드디스크에서 자원을 찾아올 수 있도록 지정이 가능하다. MCI 자원 상태 모듈(MCI resource state module)은 MCI 호출 자원을 보조 메모리에서 읽어 들여 객체 상태 분석 모듈에서 페이지/객체 상태 모듈에서 얻어진 상태 값을 이 모듈에 보내어 실제 MCI 호출 자원을 재생(playback)하거나 close시킨다. MCI 자원 재생 모듈(MCI resource play module)과 MCI 자원 닫기 모듈(MCI resource close module)은 각각 호출된 자원을 재생하고 닫아 주는 역할을 하는 모듈이다. 페이지/객체 상태 모듈(page/object state module)은 자동 프리젠테이션 되는 동안에 해당 페이지와 내부의 MIC 호출을 통해 수행되어지는 객체의 상태에 따라 상태 값을 변환시켜 이를 객체 상태 분석 모듈에 전달하여 페이지 변환 모듈에서 자동 페이지 변환이 가능하도록 상태 값을 넘겨 준다.

마지막으로 페이지 변환 모듈(page transition module)은 MCI 자원 상태 모듈과 페이지/객체 상태 모듈에서 얻어진 상태 값에 따라 이를 객체 상태 분석 모듈에서 분석하여 이 모듈에 넘기면 자동으로 페이지를 변환하는 모듈이다. 이렇게 페이지 변환은 사용자 인터페이스를 통해 다음 페이지로 이동하는 모양을 출력 값으로 내보낸다. ATS의 전체적인 모듈 구성도는 다음 (그림 10)과 같다.



(그림 9) DLL을 이용한 자동 프리젠테이션 시스템 다이어그램

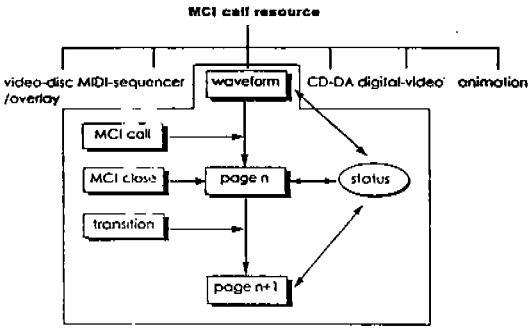
(Fig. 9) Diagram of ATS using DLL



(그림 10) ATS의 전체 구성도

(Fig. 10) Structure of ATS

따라서 ATS은 위와 같은 모듈을 통해 디지탈비디오(digital video), 비디오 오버레이(video overlay), 비디오 디스크(video disc), 미디 시퀀서(MIDI sequencer), 컴팩트 디스크 오디오(CD-DA), 애니메이션(animation), 웨이브폼(waveform) 등의 7가지 호출 자원 중 웨이브폼에 대한 상태 제어의 역할을 하게 된다. 즉 (그림 11)과 같이 웨이브폼의 상태, 페이지의 상태 등을 파악하여 웨이브폼의 메모리 적재 및 해제(release)를 수행하며 이때 페이지 전환의 기능을 동시에 수행하게 되는 것이다.



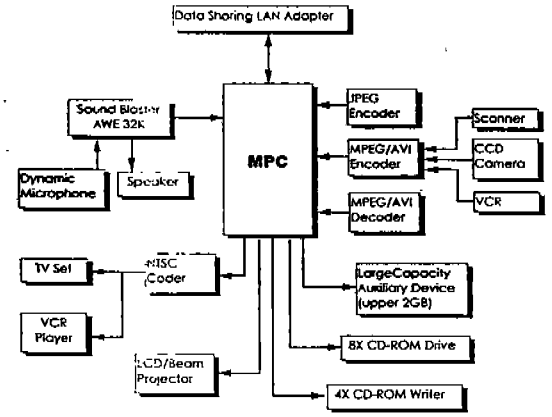
(그림 11) ATS의 역할
(Fig. 11) Role of ATS

4.2 ATS의 구현

ATS의 구현을 위한 DLL 생성은 Visual C++을 이용하여 생성하였다. 페이지 전환에 관련된 입력 이벤트가 발생하면 이를 툴북에서 제공하는 사용자 인터페이스(user interface)를 (그림 10)과 같이 보여진다. 이후 Transit.dll을 사용을 위한 선언 모듈이 Using DLL 모듈로서 각 페이지별 객체의 상태를 getStateFlag() 함수 값에 의해 추출하고 이를 객체 상태 분석 모듈(object state analysis module)을 거쳐 모든 페이지에서 호출되어지는 MCI 자원을 메모리로 적재 시킨다. 이러한 자원 적재는 MCI call 모듈에서 담당하고 각 MCI자원의 상태는 자원 상태 모듈(MCI resource state module)에서 자원의 재생 및 자원 closing을 관리한다. 적재 시킨 자원은 페이지의 상태에 따라 재생되어지고 객체 상태를 알리는 PageState() 함수의 파라미터가 set되면 자동적으로 getMCIClose() 상태를 set하여 페이지 변환 모듈(page transition module)로 이동하고 이에 따라 해당 페이지로의 이동 된 후에는 다시 getStateFlag(), getPageState(), getMCIClose() 함수값을 초기화 시켜 해당되는 변환 페이지에서 새로운 입력 이벤트를 받아들인다. 이러한 과정의 반복으로 페이지 단위의 툴북 응용 프로그램은 자동으로 첫 페이지에서 마지막 페이지까지의 자동 변환에 해당하는 navigation을 마칠 수 있게 한다.

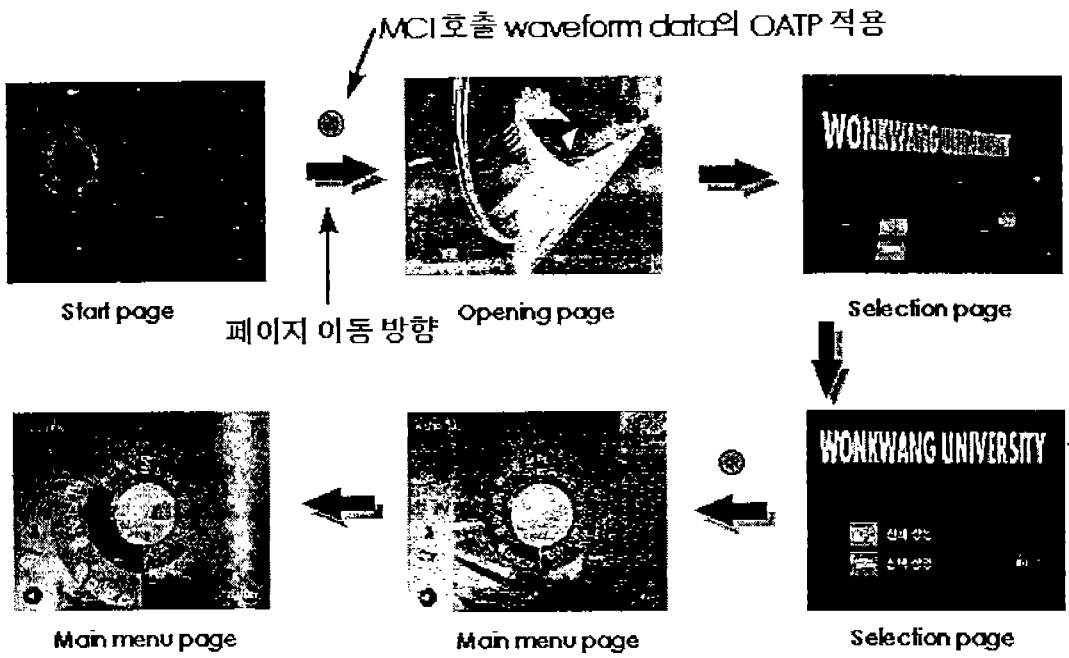
이러한 모듈을 중심으로 객체 자동 변환 알고리즘의 구현은 멀티미디어 툴북 4.0 CBT(Computer Based Training) Edition에서 수행 하였다. 따라서 자동 프리젠테이션 기능을 수행하는 ATS이 구현 되었으며 이를 멀티미디어 툴북으로의 연동이 가능하게 만들었다.

자동 프리젠테이션을 위한 모노 미디어의 수집에서부터 편집, 툴북에서의 적용에 이르는 과정에서 필요한 시스템 환경은 (그림 12)와 같다. 미디어 작업의 분할을 위한 지역 네트워크(LAN)와 미디어 캡쳐를 위한 하드웨어적인 환경을 지원하는 각종 encoder/decoder, 실제 프리젠테이션을 위한 NTSC coder와 프리젝터 등은 응용 프로그램이 완성 된 이후에 사용된다.

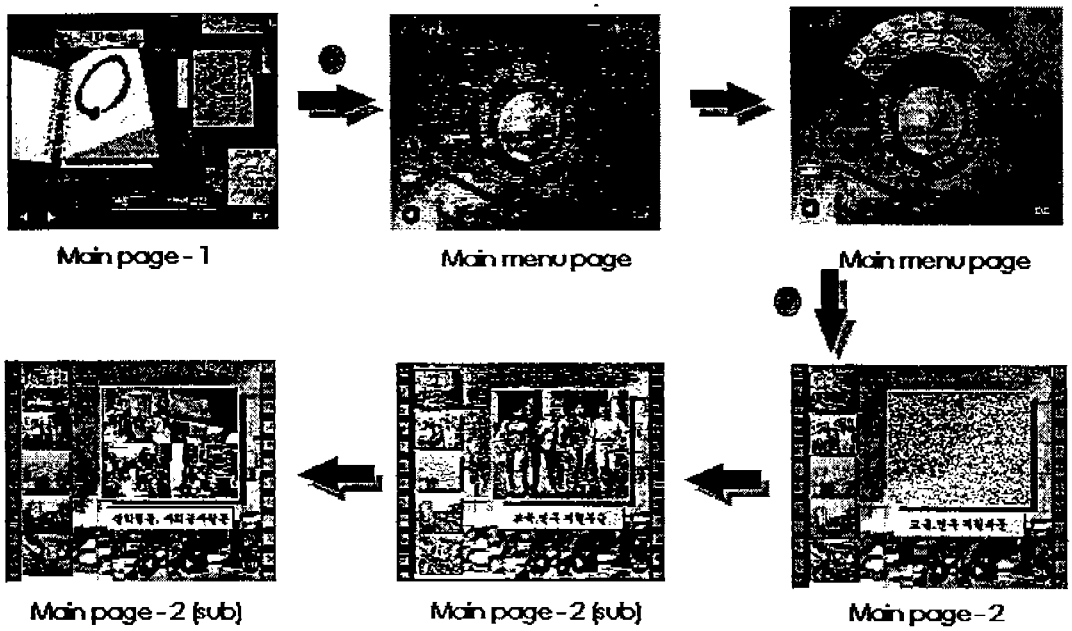


(그림 12) 구현을 위한 시스템 구성도
(Fig. 12) Map of system specification

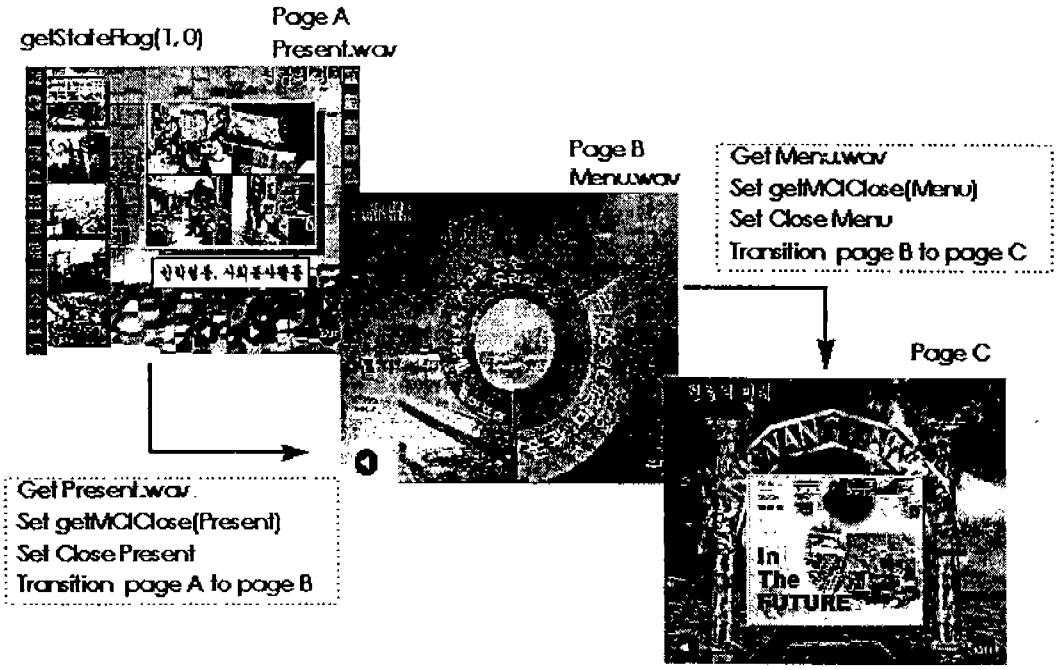
또한 ATS을 이용하여 멀티미디어 저작도구를 통한 자동 프리젠테이션 기능을 갖는 응용 프로그램에 적용해 보았다. 학교 홍보용 프리젠테이션으로 개발한 멀티미디어 기반 학교 홍보용 CD-ROM 타이틀 응용 프로그램에서 실제 MCI로 호출된 자원 중에 웨이브 폼 데이타에 대한 ATS의 적용은 (그림 13), (그



(그림 13) ATS의 적용 사례 1
(Fig. 13) Example-1 of ATS



(그림 14) ATS의 적용 사례 2
(Fig. 14) Example-2 of ATS



(그림 15) ATS의 적용 사례 3
(Fig. 15) Example-3 of ATS

림 14), (그림 15)의 적용 사례와 같이 각 화면은 멀티미디어 툴북의 페이지 단위 프리젠테이션 모양이며 화살표는 페이지의 변환을 나타내는 페이지 이동 방향을 나타낸다. 페이지의 이동 방향은 앞서 저작도구 프로그래밍 기법에서 설명한 composite 구조 형태로 되어 있다. 또한 방향 표시 위에 나타난 원은 ATS에 의한 호출 웨이브 폼 데이타의 열기/닫기 수행 및 그에 따른 상태 값의 분석에 의해 발생하는 페이지 변환이 일어나는 부분이다. (그림 15)에서의 적용 사례는 자동 프리젠테이션이 발생에 대한 ATS 메카니즘의 세부 적용 사례로서 페이지 A에서 getStateFlag(1, 0)라는 상태 값은 페이지 변환에 필요한 상태 값이 set되었으므로 페이지 A에서 open되어 있는 "present.wav"라는 이름의 MCI 호출 자원의 상태를 추출하여 이를 close시키고 페이지 A에서 페이지 B로 변환을 위한 transition이 발생시킨다. 페이지 B에서는 다시 "menu.wav"라는 이름의 웨이브 폼 데이타를 열어 동작시키다가 getStateFlag() 함수의 상태 값이 set되면 페이지 A에서와 같이 페이지 B에서 사용중인 MCI 호출 자원 "menu.wav"를 추출하고 이를 close시킨 후 페이지 B에서 페이지 C로 변환하는 transition을 발생시키는 순으로 자동 프리젠테이션이 발생한다.

5. 결 론

저작도구로 제작된 프리젠테이션 전용의 응용 프로그램에 있어서 객체 상태의 자동 변환은 필수적이며 본 연구에서 이를 위한 자동 객체 변환 알고리즘을 고안하여 이를 이용한 자동 프리젠테이션 기능을 멀티미디어 툴북에 추가하였다. 또한, 툴북의 기능 확장 방법으로 동적 연결 라이브러리를 이용하였다. 현재 상업용으로 개발되어지는 저작도구 기반 응용 프로그램의 대부분은 적절한 기능 확장을 위해 이와 같은 동적 연결 라이브러리를 사용하고 있다. 따라서 이러한 시도는 자동 프리젠테이션 기능이 미약한 멀티미디어 기반 저작도구를 보완하며 동적연결 라이브러리 형태를 취하므로 응용 프로그램의 적재 메모리 축소가 가능해졌다. 그리고 매번 객체 단위로 프

로그래밍해야 하는 번거로움을 모듈화 시킴으로써 멀티미디어 툴북의 스크립트 프로그래밍 시간의 단축 시키는 결과를 얻을 수 있다. 그러나 툴북에서는 동적 연결 라이브러리 이외에도 비주얼 베이직을 통한 기능 확장도 가능해졌다. 즉 비주얼 베이직에서 이용하는 VBX를 툴북에서 자유롭게 사용이 가능해져 활용 용도가 더욱 커지고 있다.

앞으로 본 연구결과에 의해 만들어지는 멀티미디어 응용 프로그램의 효율적인 기능 수행을 위해서는 기존의 응용 프로그램과의 상대적 평가와 MCI 자원 중에 사용되는 오디오 자원 이외의 자원인 디지탈비디오, 비디오 오버레이, 비디오 디스크, 미디 시퀀서, 컴팩트 디스크 오디오, 애니메이션 등의 자원에 대한 수용이 필요로 하며 이들 자료의 보다 간결한 사용을 위한 객체 자동 변환 알고리즘의 추가 및 보완 개발이 필요하다. 또한 클립을 통해 툴북의 응용 프로그램의 크기를 증가시키지 않고 계속적인 사용이 필요하지 않거나 동적 연결로도 응용 프로그램의 수행 저하를 발생시키지 않는 정도에서 동적 연결 라이브러리의 계속적으로 개발되어 저작도구의 기능 확장에 이용되어야 할 것이다. 또한 빠른 페이지 변환을 위해서는 멀티미디어 기반 툴북에서 사용되는 페이지나 객체에 대한 항해 인덱스 테이블(navigation index table)을 만들어 복합 구조(composite structure) 형태로 프로그래밍 되어있는 응용 프로그램의 페이지 빠른 페이지 변환이 가능하도록 해야 한다.

그리고 이들의 개발을 통해 기존의 멀티미디어 기반 저작도구에 의해 생성된 응용 프로그램과 객체 자동 변환 알고리즘을 기반으로 한 ATS 기능을 추가한 멀티미디어 기반 저작도구와의 상대적 평가가 필요하다.

참 고 문 헌

[1] Asymetrix Multimedia ToolBook 4.0 CBT Ed. -User manual, Asymetrix Corporation, 1996.

[2] Asymetrix Multimedia ToolBook 3.0, Asymetrix Corporation, 1993.

[3] Simon J. Gibbs, Dionysios C. Tsichritzis, Multimedia Programming-objects, environments and frameworks, Addison-Wesley publish, 1994.

[4] John F. Kogel Buford, Multimedia Systems, Addison-Wesley publish, 1994.

[5] Judith Jeffcoate, Multimedia in Practice-technology and application, Prentice-hall, 1995.

[6] Songbae Eun, Eun Suk No, Hyung Chul Kim, Hyunsoo Yoon, and Seung Ryoul Maeng, "Eventor:an authoring system for interactive multimedia applications," Multimedia systems, vol. 2, No. 1-6, ACM press, pp. 129-140, 1994.

[7] Barbara Beccue, "User Navigation Strategies for Multimedia Tutorials," Educational Multimedia and hypermedia, AACE, pp. 38-43, 1996.

[8] Programming with MFC and Win32-The Six-Volume Documentation Collection for Microsoft Visual C++ Version 2.0 for Win 32, Microsoft Press, 1995.

[9] Lawrence Harris, Teach yourself OLE Programming in 21 days, SAMS publishing, 1995.

[10] E. Valsky, N. Herzog, R. Peratello, W. Slany, "The Department Information System of the Information System of the Information System Department at Technical University of Vienna," Multimedia/Hypermedia in Open Distributed Environments, Proceeding of the Eurographics Symposium, Springer-Verlag Wein New York, pp. 99-102, 1994.

[11] Frans C. Heeman, Ivan Herman, Graham Reynolds, "Interaction objects the MADE Multimedia Environment," Multimedia/Hypermedia in Open Distributed Environments, Proceeding of the Eurographics Symposium, Springer-Verlag Wein New York, pp. 264-277, 1994.

[12] 류준선, *멀티미디어 동기화에 관한 연구*, 광운대학교 대학원 석사학위논문, 1992.

[13] 이만재, 박현제, 한상기, *이해하기 쉬운 멀티미디어*, 하이테크정보사, 1994.

[14] 안종길, "멀티미디어 저작도구," 정보과학회지-특집:멀티미디어, 제12권 제7호, pp. 70-82, 1994. 8.

양 옥 렬

1995년 원광대학교 컴퓨터공학과 졸업(학사)
1997년 원광대학교 대학원 컴퓨터공학과 졸업(공학석사)
1997년~현재 원광대학교 대학원 컴퓨터공학과 박사과정
관심분야:멀티미디어 데이터베이스, HCI, 멀티미디어 CAI, 멀티미디어 저작도구, 에이전트 시스템, 멀티미디어 프리젠테이션

정 영 식

1987년 고려대학교 수학과 졸업(학사)
1989년 고려대학교 대학원 전산학과 졸업(이학석사)
1993년 고려대학교 대학원 전산학과 졸업(이학박사)
1993년~현재 원광대학교 공과대학 컴퓨터공학과 교수
관심분야:컴퓨터 시뮬레이션, 분산시스템, ICAI, 멀티미디어 CAI

이 용 주

1976년 고려대학교 전자공학과 졸업(학사)
1987년 고려대학교 대학원 전자공학과 졸업(공학석사)
1992년 고려대학교 대학원 전자공학과 졸업(공학박사)
1980년~1994년 한국전자통신연구소 자동통역연구실 실장(책임연구원)
1994년~현재 원광대학교 공과대학 컴퓨터공학과 교수
관심분야:음성언어정보처리, HCI, 멀티미디어 시스템