컴퓨터 스크린 이미지와 오디오의 동영상 저작 및 재생 엔진 구현

○
황기태*, 이재문**
*한성대학교 컴퓨터 공학 전공
**한성대학교 멀티미디어 정보처리 전공

Implementation of Engine for Authoring and Playing Motion Picture of Computer Screen Images and Audio

○
Kitae Hwang*, Jaemoon Lee**
*Dept. of Computer Engineering, Hansung University
**Dept. of Multimedia Information Processing, Hansung University
e-mail : calafk@hansung.ac.kr, jmlee@hansung.ac.kr

요약

본 논문에서는 컴퓨터를 이용한 원격 강의, 원격 학습, 데모 화면 제작 등의 응용들에 필요한 동영상 멀티미디어 시스템의 설계 및 구현을 보인다. 본 논문에서 다루는 연속적으로 변하는 컴퓨터 스크린 이미지는 실세계 비디오와 크기와 영상 특성에 있어 차이점을 가지며 기존의 MPEG 등과 같은 압축 알고리즘이 부적합하다. 시간적으로 변하는 컴퓨터 스크린과 컴퓨터에서 발생하는 오디오로 구성되는 동영상을 저작 재생하는 멀티미디어 시스템 구현 내용과 시스템 성능 평가 결과를 보인다.

1. 서론

동영상 기술의 핵심은 각종 미디어의 압축 성능과 미디어들 사이의 동기화 정확성을 이루는데 있다[7,8,9,12]. 대부분의 동영상 기술은 비디오 카메라 등에 의해 다루어지는 실세계의 영상에 적용되어 왔다. 그러나 시간적으로 변하는 컴퓨터 스크린에 대해서도 동영상 기술이 적용된다면 효과적일 수 있다. 특정 프로그램의 사용 설명, 프로그램의 실행 과정이나 실행 결과를 동적으로 저장하거나, 소프트웨어 패키지의 사용 방법 등은 보통 문서 형식으로 작성된다. 이러한 문서는 사용자가 보고 학습하기에 많은 시간과 노력이 요구되며 때로는 이해하기 어려운 면이 있다. 이 문서들을 동영상으로 만들어 배포한다면 매우 효과적일 것이다[4,5]. 또한 컴퓨터를 이용한 원격 강의, 원격 수강, 강의 노트 제작, 데모 화면 제작 등 많은 응용 분야가 존재한다.

한편 시간적으로 변하는 컴퓨터 스크린의 이미지는 실세계 비디오 이미지와는 다른 몇 가지 차이점을 가지고 있다. 첫째 모니터 기술의 발전과 가격 저렴화로 인해 대부분의 응용 소프트웨어는 24 비트 칼라와 1024*768 이상의 해상도로 동작된다. 24 비트 칼라에 1024*768 의 해상도를 가지는 한 화면은 약 2.25MB 의 데이터를 차지한다. 이는 MPEG1 화면의 최대 크기인 360*240 에 비해 9 배나 큰 데이터 량이다[13]. 그러므로 컴퓨터 스크린 전체를 동영상으로 구성하는 경우 데이터량은 매우 커지게 된다. 두번째 특징은 실세계 영상은 화면 전반에 걸쳐 적은 양이지만 지속적으로 화면이 변한다. 그러므로 실세계 영상이 자연스러운 연속적인 화면이 되려면 초당 15 프레임이상 샘플링되어야 한다[7]. 그러나 응용 프로그램이 실행되고 있는 컴퓨터 화면은 본 논문의 실험 결과 초당 2 번 정도의 샘플링이면 충분한 것으로 평가된다. 세번째 특징은 컴퓨터 화면의 변하는 부분이 국소적이라는 특징이 있다. 예를 들면 워드 프로그램 사용시 주메뉴를 선택하면 해당하는 메뉴 부분이 사각형의 영역에서 변한다.

이와 같은 이유로 실세계의 동영상을 압축하는 주된 기법인 MPEG[10,11]은 시간적으로 변하는 컴퓨터 스크린 이미지를 동영상화하는데 적합하지 않다. 따라서 본 논문은 시간적으로 변하는 컴퓨터 스크린 이미지와 컴퓨터에서 발생하는 오디오 스트림를 동시에 녹화하여 동영상으로 제작하고 이를 다시 재생하는 멀티미디어 시스템의 설계 구현한 CAM 시스템을 보인다.

본 논문은 다음과 같이 구성된다. 2 장에서는 본 논문에서 설계 구현한 CAM 시스템의 구성을 소개하고, 3 장에서는 CAM 시스템의 엔진의 설계 구현된 내용을 보이며, 4 장에서는 CAM 시스템의 성능을 평가하고 5 장에서 결론을 맺는다.

2. CAM 시스템 구성

본 논문에서는 제안하는 시스템은 시간적으로 변하는 컴퓨터 스크린 이미지와 컴퓨터에서 발생하는 모든 오디오로 구성되는 멀티미디어 스트림을 다룬다. 이 두 연속적인 스트림은 압축되어 멀티미디어 파일로 저장한다. 본 논문은 다음과 같은 설계 원칙을 가진다.

- 시간 연속적인 스트리밍
- 스크린과 오디오 스트림의 동기화
- 데이터의 소량화
- 마우스의 원활한 움직임 구현

CAM 시스템은 마이크로소프트사의 DirectShow[3]라는 패키지를 이용하여 설계 구현되었으며 그림 2-1 과 같이 구성된다. 필터는 하나의 COM[6] 모듈이며, 이들이 파이프라인처럼 연결되어 한 필터에서 입력 데이터를 처리한 후 다음 필터로 넘겨주는 식으로 구성된다. 마이크로소프트에서 제공되는 오디오 압축/인코딩 필터인 GSM 610 필터와 Audio Render 필터를 제외한 나머지 필터는 실제 구현하였다. 구현된 각 필터의 기능은 표 2-1 과 같다.

CAM 시스템은 저작된 멀티미디어 데이터의 저장을 위해 ASF 포맷을 이용한다. ASF[1] 파일 포맷은 늘리 사용되는 멀티미디어 파일 포맷으로 많은 멀티미디어 파일들이 ASF 파일 포맷으로 구현되었다. 마이크로소프트사는 최근에 ASF 를 발전시킨 윈도우 미디어 파일 포맷[2]을 개발하여 사용하고 있다.

윈도우 미디어 파일 포맷은 ASF 파일 포맷과 거의 동일하나, 미디어 데이터가 반드시 MPEG4 로 인코딩 되어야 한다는 특징을 가진다. 서론에서도 밝혔듯이 시간적으로 변하는 스크린 이미지의 특성상 MPEG 코딩이 적합하지 않기 때문에 새로운 압축 방식으로 멀티미디어 파일을 생성하기 위해서는 윈도우 미디어 파일 포맷 대신 ASF 파일 포맷을 사용한다. ASF 파일 포맷의 입출력을 위해서는 마이크로소프트사의 ASF PDK[1] 라이브러리를 사용하였다. ASF 파일은 헤더와 미디어 스트림 데이터가 담긴 데이터 유닛들로 구성된다. 그러므로 ASF 파일의 기본 입출력 단위는 하나의 데이터 유닛이다. 데이터 유닛의 크기는 개발자에 의해 결정된다.

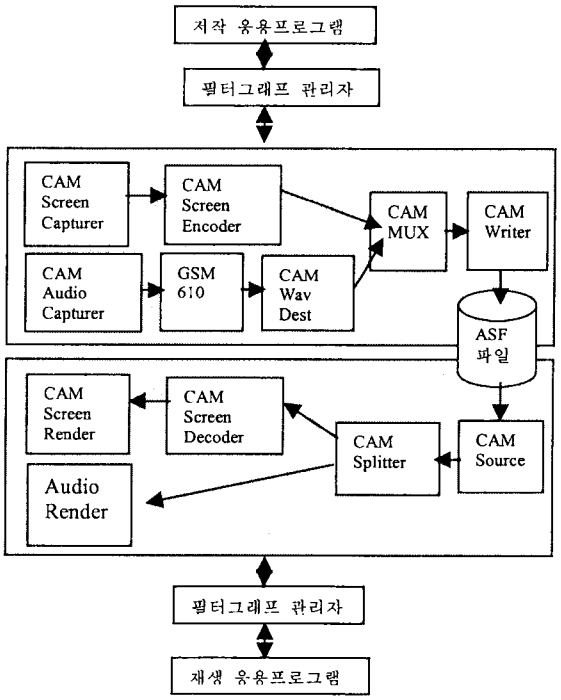

그림 2-1 CAM 시스템 구성

표 2-1 필터의 기능 요약

필터 이름	기능
CAM Screen Capturer	스크린을 캡쳐하고 이전 스크린에서 변경된 부분 추출
CAM Screen Encoder	변경된 스크린 이미지 압축

CAM Mux	오디오와 스크린 이미지 멀티플렉싱
CAM Writer	압축된 오디오/스크린 스트림을 ASF 포맷으로 저장
CAM Audio Capturer	오디오 소스로부터 샘플링
GSM 610	샘플링된 오디오를 GSM 포맷으로 인코딩
CAM Wav Dest	GSM 포맷의 오디오 스트림을 wav 포맷으로 변환
CAM Source	ASF 파일로부터 스트림 읽기
CAM Splitter	오디오인지 스크린 스트림인지 분별
CAM Screen Decoder	압축된 스크린 스트림을 복원
CAM Screen Render	복원된 스크린 스트림을 화면에 출력
Audio Render	오디오의 디코딩과 동시에 재생

3. 저작 및 재생 엔진 구현

3.1 구성

그림 2-1 에 묘사된 CAM 시스템 구성도에서 스트리밍 엔진은 구체적으로 그림 3-1 과 3-2 와 같다. 그림 3-1, 3-2 에서 T 는 스레드를 의미하며, B 는 버퍼를 의미한다. 필터와 필터 사이에는 버퍼가 존재하며 한 필터에서 다운스트림 필터로 데이터를 넘길 때 이용된다.

3.2 저작 엔진

멀티미디어의 소스로부터 미디어 스트림을 받아 이를 멀티미디어 파일에 저작하는 과정을 실행하는 엔진을 저작 스트리밍 엔진이라고 부른다. 이 엔진은 그림 3-1 과 같이 설계되었다.

T1 스레드는 주기적으로 스크린 이미지를 캡쳐하여 버퍼 B0 에 저장한다.

T2 스레드는 B0 에 캡쳐된 스크린 이미지를 이전 스크린 이미지와 비교하여 변경된 부분을 B1 버퍼로 전송하고 CAM Screen Encoder 필터의 코드를 실행하여 B1 의 내용을 압축하여 다시 B2 버퍼에 저장한다. 그리고 나서 CAM Mux 의 코드를 실행하여 압축된 이미지를 B2 에서 B3 로 전송하며 OutputQ 인 큐에 요청을 삽입하고 리턴한다. T4 스레드는 오디오 소스로부터 오디오를 캡쳐하고 B4 버퍼에 저장하고 다시 GSM610 필터의 코드를 실행하여 압축/인코딩한 후 B5 버퍼에 저장한다. 그리고 나서 CAM Wav Dest 필터의 코드를 실행하여 오디오 스트림을 웨이브(wav) 포맷으로 변환한 후 B6 버퍼에 삽입한다. 다시 CAM Mux 필터의 코드를 실행하여 B3 에 전송하며 OutputQ 큐에 요청을 삽입하고 리턴한다. CAM Mux 의 코드는 T2 와 T4 스레드에 의해 동시에 진입이 가능하므로 임계 영역(critical section)이 설정되었다.

T3 스레드는 압축/인코딩된 스크린 이미지와 오디오 스트림을 ASF 파일 포맷으로 변환하여 저장하는 일을 수행한다.

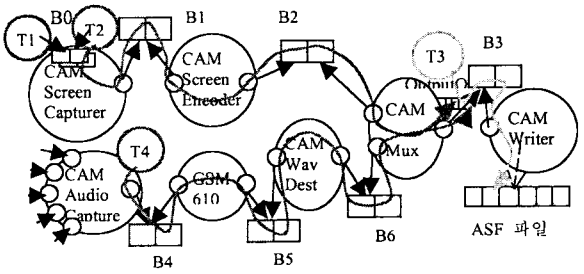

그림 3-1 저작 스트리밍 엔진

3.3 재생 엔진

멀티미디어 파일로부터 각 스트림 데이터를 읽고 이를 재생 장치에 출력시키는 작업 엔진을 재생 스트리밍 엔진이라고 부르며 그림 3-2 와 같이 묘사된다. ASF 파일의 재생을 위해서 재생 스트리밍 엔진은 4 개의 스레드로 구성된다. T1 스레드는 일단 하나의 스트림 데이터 유닛을 ASF 파일로부터 버퍼 B1 으로 읽어들이고 CAM Splitter 필터의 코드를 실행하여 어떤 스트림인지 판별하고 스크린 스트림이면 B2 버퍼에, 오디오 스트림이면 B3 버퍼에 삽입한다.

T2 스레드는 B2 버퍼로부터 스트림 데이터를 읽고 압축 복원을 실행하기 위해 CAM Screen Decoder 필터의 코드를 실행한다. 압축복원된 스크린 이미지를 B4 에 삽입하고 역시 요청을 해당하는 OutputQ 큐에 삽입한다.

T3 스레드는 B3 버퍼로부터 오디오 데이터를 디코딩하여 재생하기 위해 Audio Render 필터의 코드를 실행한다.

T4 스레드는 B4 버퍼로부터 압축복원된 스크린 이미지를 스크린에 출력하기 위해 CAM Screen Render 필터의 코드를 실행한다. 압축 복원된 스크린

이미지는 현재 스크린에 출력되어 있는 이미지에서 변경된 부분만을 가지므로 T4 스레드는 B4 버퍼에 있는 스크린 이미지의 재생 시작 시간이 될 때까지 기다린 후 스크린에 출력한다.

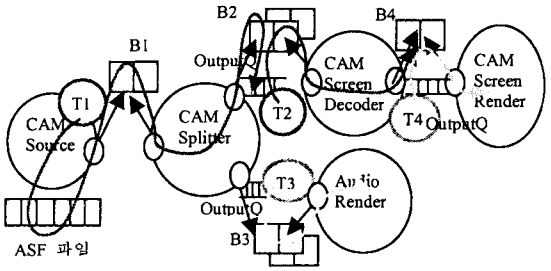

그림 3-2 재생 스트리밍 엔진

3.4 엔진의 멀티스레드 구현 내용

멀티스레드를 설계하는 기본 원칙은 각 스트림의 동기화와 처리 시간의 최소화에 있다.

다음은 저작 스크리밍을 위한 멀티스레드 설계의 주요 내용을 설명한다.

(1) 스크린 캡처와 압축을 분리하여 처리

스크린 캡처 작업은 주기적으로 이루어져야 하는 작업이며 압축은 스크린 데이터의 내용에 따라 처리 시간에 차이가 나므로 이 두 작업을 별도의 스레드로 처리하도록 설계하였다.

(2) 오디오 캡처 및 인코딩은 하나의 스레드로 처리 가능하다.

오디오 인코딩 처리에 따른 시간은 매우 작으며 wav 형식으로 변환하는 작업 역시 매우 짧은 시간 밖에 요하지 않기 때문에 하나의 스레드로 처리하여도 문제가 없다.

(3) 디스크 입출력에 스레드를 별도 할당

T3 스레드는 스트림 데이터를 ASF 파일 포맷으로 변환하고 디스크 쓰기를 실행한다. 디스크 쓰기는 운영체제의 의해 물리적으로 처리되므로 상황에 따라 쓰기 시간이 달라지므로 디스크 쓰기를 실행하는 스레드는 항상 균일한 처리 시간을 보장받을 수 없다. 그러므로 이를 별도 스레드로 생성하여 처리하는 것이 효과적이다.

재생 스트리밍을 위한 멀티스레드 설계의 주요 내용을 설명한다.

(4) ASF 파일로부터 스트림 읽기와 압축복원은 별도의 스레드로 분리

T1 스레드는 ASF 파일로부터 스트림 데이터를 계속적으로 읽어 내어 버퍼 B2 에 저장하는 일을 수행한다. 만일 T1 스레드가 스트림 데이터를 파일로부터 읽은 후 스크린 스트림의 압축복원까지 실행한다면 그 동안 오디오 스트림의 공급이 일시 중단 되므로 소리가 끊기게 되는 현상이 일어나게 된다. 혹은 오디오와 스크린의 동기가 일치하지 않게 된다. 이를 해결하기 위해 T2 와 T3 스레드를 T1 스레드와 별도의 스레드로 구성하였다.

(5) 스크린 스트림의 압축복원과 재생을 서로 분리된 스레드로 처리한다.

각 스트림의 데이터 유닛은 재생 시간에 대한 정보를 가지고 있으며 스트림 데이터가 재생되기 전에 재생 시작 시간까지 기다려야 한다. 만일 T2 스레드가 스트림 데이터의 디코딩 후 렌더링(재생)까지 실행한다면, T2 스레드는 스트림 데이터에 포함한 재생 시작 시간까지 기다리게 되므로 현재 B2 버퍼에서 압축 복원을 기다리는 다음 스트린 이미지를 늦게 복원하게 된다. 즉, B2 에 존재하는 다음 스크린 이미지는 자신의 재생 시작 시간 전까지 복원이 이루어지지 못하게 되는 문제가 발생하게 된다. 이렇게 되면 재생 시작 시간 이후에 재생되는 스크린 이미지가 존재하게 되며, 이때 오디오 스트림과 동기가 일치하지 않게 되는 문제를 야기시킨다. 그러므로 T4 스레드를 독립적으로 생성하여 렌더링 만을 담당하게 한다.

3.5 마우스의 움직임 캡쳐

사용자가 컴퓨터를 사용함에 있어 마우스 사용의 빈도수는 대단히 높다. 스크린에 변화가 없어도 마우스의 움직임은 결국 스크린이 변한 것으로 판단하게 하므로 마우스의 움직임을 따로 추적하여 관리하면 압축되는 스크린 스트림의 양을 줄일 수 있다. 마우스의 위치를 추적하는 스레드를 따로 두고 화면 캡쳐보다 높은 빈도수로 마우스의 위치를 계속 추적한다. 마우스 모양은 응용 프로그램, 클릭,

드래깅 등에 따라 계속 변하기 때문에 마우스의 위치 추적과 마우스의 모양 정보를 함께 알아낸다. 마우스에 관한 이 정보들은 ASF 파일로 보내져 스크린 스트림의 일종으로 저장된다.

3.6 스크린 이미지의 압축

시간적으로 변하는 스크린 스트림을 압축하기 위해 MPEG 등에서 사용되는 기본 방법을 변형한 것으로 캡쳐된 이전 스크린과 현재 캡쳐한 스크린의 변화된 부분만을 가려내고 이를 JPEG 으로 압축하는 방법을 이용하였다.

4. 성능 평가

실험에 사용된 컴퓨터는 윈도우 NT 운영체제가 탑재된 펜티움 II 300 MHz, 메인 메모리 64MB 의 일반적인 PC 이다.

표 4-1 은 3 가지 작업 부하에 대해 CAM 시스템을 사용하여 저작한 멀티미디어 데이터량(ASF 파일 크기)을 평가한 결과를 보여준다. 본 논문에서는 구체적으로 밝히지 않지만 기존에 존재하는 Lotus 의 Screen CAM[4]을 이용한 경우, 분당 10MB 정도가 되는 것으로 측정되었다. 본 시스템은 대략 1MB/min 의 압축 성능, 즉 0.13Mb/sec 의 데이터 율로서 10Mb/sec 의 LAN 환경에서 실시간으로 저작 및 재생이 가능함을 의미한다. 그리고 본 실험은 1024*768 의 전체 화면을 캡쳐한 경우이므로 저작하고자 하는 영역이 스크린의 일부인 경우에는 더 낮은 데이터 율을 얻을 수 있다.

표 4-1 성능 평가 결과(오디오: 44.1KHz, 16 비트, 스테레오)

JPEG 압축질	측정시간	작업부하	MB/min
10%	1 분 46 초	flash	0.68
	16 초	avi	1.15
	38 초	한글워드	0.71
30%	1 분 46 초	flash	0.73
	16 초	avi	1.27
	38 초	한글워드	0.57
75%	1 분 46 초	flash	0.85
	16 초	avi	1.53
	38 초	한글워드	0.74

5. 결론

본 논문에서는 컴퓨터를 이용한 원격 강의, 강의 노트 제작, 데모 화면 제작, 소프트웨어 패키지 사용 방법 등의 응용들에 필요한 동영상 멀티미디어 시스템을 제안하였다. 이들 응용들은 실세계 비디오와는 달리, 시간적으로 변하는 컴퓨터 스크린 스트림과 오디오 스트림으로 구성되는 동영상 시스템을 필요로 한다. 본 논문에서 구현된 멀티미디어 시스템은 약 0.13Mb/sec 내외의 데이터 율을 보이는 것으로 평가되어 매우 성능이 우수한 것으로 평가되며 현실적으로 실용 가능한 것으로 평가된다.

참고문헌

[1] Microsoft, Advanced Streaming Format PDK, 1999

[2] Microsoft, Windows Media Format SDK, 1999

[3] Microsoft, Microsoft DirectShow SDK, 1999

[4] Lotus, "ScreenCAM", http://www.lotus.co.kr

[5] 미리온 시스템즈, Wincam 2000, http://www.wincam.net

[6] Jonathan Bates, Creating Lightweight Components with ATL, SAMS

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

[8] D. P. Anderson and G. Homsy, " A Continuous Media I/O Server and its Synchronization Mechanism, " IEEE Comupter Special Issue on Multimedia Information System, pp. 51-7, Oct. 1991

[9] T. D. C. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand, " IEEE Multimedia, Vol. 1, No. 3, 1994

[10] D. L. Gall, "MPEG: A Video Compression Standard for Multimedia Applications, " Comm. of the ACM, Vol. 34, No. 4, 1991, pp. 46-58.

[11] F. Back, A. T. Lindsay, "Everything you wanted to know about MPEG-7 ," IEEE Multimedia, 1999, pp. 65-77.

[12] D. D. Kandlur, M. S. Chen, and Z. Y. Shae, "Design of a Multimedia Storage Server, " IBM Rsearch Report, Jun 1993

[13] 정제창, "최신 MPEG, " 교보문고, 1995