DirectShow 프로그래밍을 위한 C 소스 코드 자동 생성 기법

(Automatic C Source Code Generation Technique for DirectShow Programming)

동 지 연 † 박 선 화 † 엄 성 용 ††
(Ji-Youn Dong) (Sun-Hwa Park) (Seong-Yong Ohm)

요 약 본 논문에서는 DirectShow 프로그래밍의 주요 개발 도구인 그래프 에디터에서 작성된 필터 연결 그래프로부터 C 소스 코드를 자동 생성하는 시스템에 대해 설명한다.

기존의 DirectShow 프로그래밍 환경에서는 그래프 에디터를 이용한 프로그램 설계 및 실행 확인 작업과 실제 프로그램 코드를 작성하는 프로그램 개발 작업이 별도로 이루어진다. 이에 반해, 본 시스템을 사용할 경우, 멀티미디어 응용 프로그램 개발자는 소스 코드를 직접 일일이 수정할 필요 없이, 그래프 에디터를 이용하여 필터 삽입 및 필터 연결을 통한 프로그램 설계 작업을 수행한 다음, GRF 파일로 저장하기만 하면, 원하는 C 소스 프로그램을 자동적으로 얻을 수 있기 때문에 보다 효과적이고 훨씬 신속한 DirectShow 프로그래밍이 가능하다.

더욱이 본 시스템은, 고정된 개수의 매우 제한된 미디어 제어 기능만을 소스 코드에 추가할 수 있는 기존의 시스템과는 달리, 시스템 사용자인 프로그램 개발자로 하여금 자신이 개발하고자 하는 응용 프로그램에 추가할 미디어 제어 기능을 보다 쉽고 다양하게 선택할 수 있도록 지원하기 때문에 보다 실용적인 도구로 활용될 수 있다.

키워드 : DirectShow, 그래프 에디터, GRF, 필터, 자동생성시스템, 사용자 인터페이스

Abstract In this paper, we present an automatic C source code generation system for DirectShow based multimedia application programming. In this system, C source code is automatically synthesized from the filter connection graph edited with GraphEdit, a utility tool provided with DirectShow SDK package from Microsoft.

In traditional DirectShow programming environments, program design and brief testing steps are usually done with GraphEdit tool just by inserting filters and connecting them properly, while actual implementation of the program should be done separately. The filter connection graph information from GraphEdit is used just as a reference in such the implementation step. Therefore, our system which automatically generates C source code directly from the filter connection graph of GraphEdit seems very useful and many programmers can develop DirectShow based multimedia application programs more effectively and quickly using our system.

In addition, our system supports more various media stream control functions for the generated application programs than the existing system such as Wizard which supports limited and fixed number of media control functions only. This feature allows more flexibility in the user interface of the generated source program and makes our system more practical for DirectShow based programming.

Key words : DirectShow, GraphEdit, GRF, filter, source code generation, user interface

· 본 연구는 서울여자대학교 2003학년도 교내특별연구비의 지원을 받아 수행되었습니다.

† 학생회원 : 서울여자대학교 컴퓨터학과
dongji79@hanmail.net
bban@swu.ac.kr

†† 종신회원 : 서울여자대학교 정보통신공학부 교수
osy@swu.ac.kr

논문접수 : 2003년 8월 7일
심사완료 : 2004년 1월 19일

1. 서 론

마이크로소프트(Microsoft)사의 DirectX[1]는 마이크로소프트 윈도우 운영체제에 구축되어 있는 멀티미디어 처리용 API(Application Programming Interfaces)의 향상된 기능으로서 PC 상에서 윈도우용 멀티미디어 응용 프로그램을 개발하는 표준 개발 플랫폼을 제공한다. DirectX의 일부인 DirectShow[2]는 기존의 Vfw(Video for Window)와 HCI(Human Computer Interface)를 대체하는 새로운 Audio/Video 지원 API로서, 멀티미디어와 관련된 다양한 유틸리티들을 제작할 수 있는 그래프 에디터(GraphEdit)[3]를 비롯하여 멀티미디어 응용 프로그램의 개발에 필요한 여러 가지 유틸리티들과 다양한 예제 코드 등을 제공하고 있다.

그림 1은 DirectShow의 그래프 에디터를 사용하여 멀티미디어 관련 응용 프로그램을 개발하는 과정에 대한 일반적인 작업 흐름도를 보여준다. 이 방법에서 멀티미디어 응용 프로그램 개발자는 우선 DirectShow의 그래프 에디터를 이용하여 '필터(filter)'라고 부르는 DirectShow 컴포넌트들을 잘 선택하고, 이 필터들의 입출력 핀들을 적절히 연결하는 방식으로 프로그래밍 작업을 수행한다. 그런 다음, 그래프 에디터에 제공되는 실행 버튼을 이용하여 직접 수행시켜 봄으로써 필터 선택 및 연결이 올바르게 되었는지를 확인해 본다.

이러한 방식의 작업은 매우 빠른 시간 내에 실행 결과를 얻어 볼 수 있는 장점이 있다. 그러나 그래프 에디터는 작성된 필터 연결 정보에 의해 수행되는 실행 결과를 보여주긴 하지만, 이에 해당하는 실행 프로그램이나 소스 프로그램을 별도로 생성해 주지는 않는다. 따라서 응용 프로그램 개발자들은 그래프 에디터를 이용한 설계 및 동작 확인 작업이 끝나면, 작성된 필터 연결 그래프를 토대로 별도의 프로그래밍 작업을 따로 수행해야 하는 불편을 겪고 있다. 따라서 이러한 문제점을 해결하기 위해서는 작성된 필터간의 연결 그래프로부터 직접 프로그램 소스 코드를 자동 생성하는 기법이 필요하다.

이에 본 논문에서는 이러한 점들을 보완한 새로운 DirectShow 프로그래밍용 C 소스 코드 자동 생성 시스템을 소개한다. 이 시스템에서는 그래프 에디터의 필터 연결 그래프로부터 C 소스 코드를 자동 생성하는 한편, 시스템 사용자인 프로그램 개발자로 하여금 자신이 개발하고자 하는 응용 프로그램에 추가할 미디어 제어 기능을 보다 쉽고 다양하게 선택할 수 있도록 지원한다. 이를 위해, 주어진 필터 연결 그래프에 사용된 각 필터의 속성을 분석하여 개발될 응용 프로그램에 적용할 수 있는 기본적인 미디어 제어 기능, 즉 응용 프로그램에서 미디어 제어 버튼으로 만들어 실행할 수 있는 기본 제어 기능들을 모두 추출하여 프로그램 개발자에게 보여주고, 개발자가 이 기능들 가운데 응용 프로그램에 추가할 기능(들)을 선택하면, 이렇게 선택된 기능을 수행하는 미디어 제어용 소스 코드를 이미 생성된 소스 코드에 추가한다.

그림 2는 본 논문에서 제안하는 새로운 방식의 작업 흐름도를 보여준다. 이 방식에서는 개발자는 그래프 에디터를 이용하여 필터 연결 그래프를 작성하여 GRF 파일로 저장한 후, 이를 소스 코드 자동 생성 시스템의 입력으로 준다. 소스 코드 자동 생성 시스템이 동작하면, 주어진 GRF 파일 및 시스템 내부 파일인 '필터 속성

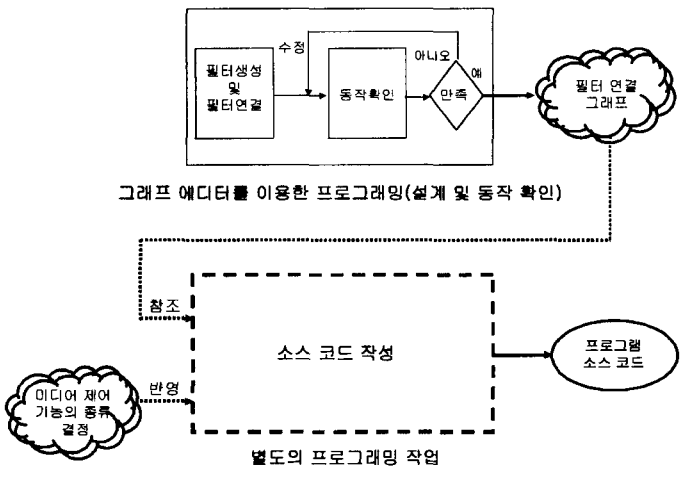

그림 1 DirectShow 프로그래밍을 위한 기존 방식의 작업 흐름도 예

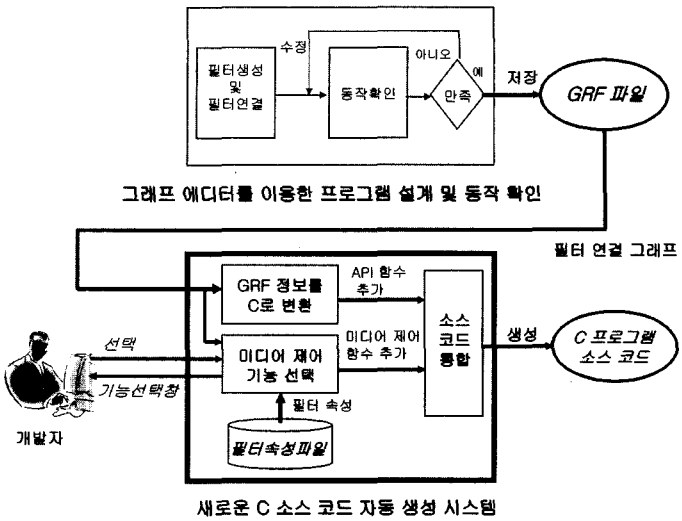

그림 2 소스 코드 자동 생성 시스템을 이용한 프로그래밍 작업 흐름도 예

파일'에 대한 분석을 통해, 적용 가능한 미디어 제어 기능의 목록을 나열하여 기능 선택 창을 통해 개발자에게 보여주는데, 이 때, 개발자는 단지 소스 코드에 들어갈 미디어 제어 기능(실행 버튼)의 종류를 선택하면, 이러한 것들을 모두 반영한 C 소스 코드가 자동 생성된다. 이러한 방식으로 멀티미디어 응용 프로그램을 개발할 경우, 개발자는 소스 코드를 전혀 수정할 필요가 없을 뿐 더러, 수작업에 따른 코드 작성시 동반되는 오류를 줄일 수 있으며, 필터 연결 그래프로부터 직접 자신이 원하는 C 소스 프로그램을 훨씬 빠른 시간 내에 얻을 수 있기 때문에, 그림 1의 기존 방식에 비해 보다 효과적인 프로그램 개발이 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 연구에 대해 소개하고, DirectShow 프로그래밍의 주요 도구인 그래프 에디터의 기능 및 본 논문 전체에 걸쳐 사용할 한 샘플 예제를 소개한다. 3장에서는 그래프 에디터에 의해 저장된 필터 연결 그래프로부터 소스 프로그램을 자동 생성하는 방법에 대해 각 단계별로 설명한다. 특히, 기존 연구와 차별되는 부분으로서 다양한 미디어 제어 기능을 추출하고 선택하여, 이에 대한 미디어 제어용 소스 코드를 생성하는 방법에 대해 자세히 설명한다. 4장에서는 구현된 시스템을 이용하여 소스 코드를 생성한 결과를 토대로 기존 연구와의 장단점을 비교한다. 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

2.1 기존 연구

프로그램 소스 코드를 자동 생성하는 기존 연구로는 다양한 기법들이 이미 존재한다[4-8].

참고문헌 [4]와 [5]에서는 프로그래밍 개발시 미리 정의된 디자인 패턴 요소를 효율적으로 적용할 수 있도록 디자인 패턴을 소스 코드 자동 생성시 활용한다. 참고문헌 [6]에서는 여러 CASE 도구에서 사용하는 각종 저장 양식으로부터 CASE 도구간의 정보 교환 표준 양식인 CDIF(CASE Data Interchange Format) 자료를 자동 생성한다. 그리고 참고문헌 [7]의 EJB 컴포넌트(Enterprise Java Beans Component)용 소스 코드 생성 기법은 웹 환경 하에서 요구되고 있는 컴포넌트 기반의 소프트웨어 개발(Component Based Development) 환경을 지원한다. 그러나 이러한 시스템들은 너무 일반적이거나 특정한 환경에 적합한 소스 코드 생성 기법으로서 DirectShow용 프로그래밍 개발 환경에 이를 직접 적용하는 데 어려움이 있다.

한편, DirectShow 프로그래밍을 위한 소스 코드 자동 생성 시스템으로는 위저드(Wizard) 시스템[8]이 있다. 이 시스템은 그래프 에디터의 결과로부터 직접 C 소스 코드를 생성하는 도구이지만, 코드 생성시 항상 Play, Stop, Pause 등의 간단한 미디어 스트림 처리 기능만을 고정적으로 제공하기 때문에, 개발할 프로그램이 Step Forward, Step Backward, Fast Forward, Rewind 등을 포함한 다양한 제어 기능을 갖추려면, 별도의 프로그래밍을 통해 자동 생성된 소스 코드를 다시 일일이 수정해 주어야 하는 단점이 있다.

2.2 그래프 에디터의 주요 기능

DirectShow의 그래프 에디터는 멀티미디어 관련 유틸리티들을 제작할 수 있는 아주 유용한 도구로서 필터

라고 불리는 DirectShow 컴포넌트들을 다양한 방법으로 구성하여 테스트해 볼 수 있는 기능을 제공한다.

그림 3은 'DVD 네비게이터(Navigator)'라는 예제 프로그램을 그래프 에디터를 이용하여 작성한 예이다. 이 예의 경우, 6개의 필터와 6개의 필터 입출력 연결로 구성되어 있다. 이 예에서 가장 중요한 역할을 수행하는 DVD 네비게이터 필터는 DVD 타이틀의 내용을 받아들여 이를 다양한 방법으로 실행시켜 주는 소스 필터(Source Filter)로서, 이 필터를 통해 전달된 DVD 음향은 오디오 디코더(CyberLink Audio Decoder)와 사운드 디바이스(Default DirectSound Device) 등의 필터들을 통해 출력되고, DVD 영상은 오버레이 믹서 (Overlay Mixer) 및 비디오 디코더(CyberLink Video/SP Decoder), 비디오 렌더러(Video Renderer) 등의 필터들을 통해 출력된다.

각 필터간의 입출력은 필터간의 핀 연결로 표현되는데, 연결되는 핀의 형식은 반드시 일치되어야 한다. 필터의 입출력 핀을 연결하는 방법에는 자동 연결 방법과 수동 연결 방법이 있다. 자동 연결 방법에서는 처음 나오는 소스 필터에 대해 마우스의 오른쪽 버튼을 누를 때 나오는 서브 메뉴에서 <Render Pin>을 선택한다. 그러면, 소스 필터의 해당 핀과 일치하는 핀 형식을 가진 필터가 자동 삽입되어 연결된다. 한편, 수동 연결 방법에서는 창 위의 <Graph> 메뉴의 <Insert Filter> 메뉴에서 해당 필터를 찾아 직접 연결하는데, 이 경우, 앞의 필터에 출력 핀 형식과 새로 추가하는 필터의 입력 핀 형식이 서로 맞지 않으면 연결되지 않으므로 형식에 맞는 필터를 잘 선택해서 연결해 주어야 한다.

필터 연결 그래프의 편집이 완료되면, 그래프 에디터 화면상에 있는 실행 버튼 아이콘(▶)을 눌러 프로그램이 개발자 자신이 원하는 데로 동작하는지 확인할 수 있다. 그림 3의 예의 경우, 소스 필터로 DVD 네비게이터를 사용했기 때문에 DVD-ROM에 들어있는 DVD 타이틀이 직접 플레이(Play)된다. 그림 4는 그림 3의 예에 대한 실행 화면을 보여준다. 이와 같이, 그래프 에디터를 이용하여, 각종 미디어 소스에 대해 다양한 필터 연결을 구성하여 이를 실행해 봄으로써 동작 여부를 쉽게 직접 확인해 볼 수 있다.

그래프 에디터에서 작성된 필터 연결 그래프는 FGML(Filter Graph Markup Language) 양식 또는 GRF(GRaphedit File) 양식으로 저장될 수 있다[2]. 확장자가 .XGR인 FGML 파일은 XML(Extensible Markup Language) 형태의 텍스트 파일이기 때문에 읽기에 편하고 정보 호환성 면에서 유리한 면이 있으나, XML 표준을 완벽히 만족하지 않는다. 그래서 DirectX 8.1 SDK[2]에서는 장기적으로 볼 때 이 양식을 참고용으로 사용하는 것은 무방하지만, 보관용으로는 이 양식을 사용하지 말 것을 권고하고 있다. 반면에, GRF 파일은 이진 파일이어서 정보 추출이 다소 복잡하고 다른 도구와의 호환성이 떨어지는 문제점은 있으나, 그래프 에디터

그림 4 그래프 에디터를 통한 실행 예

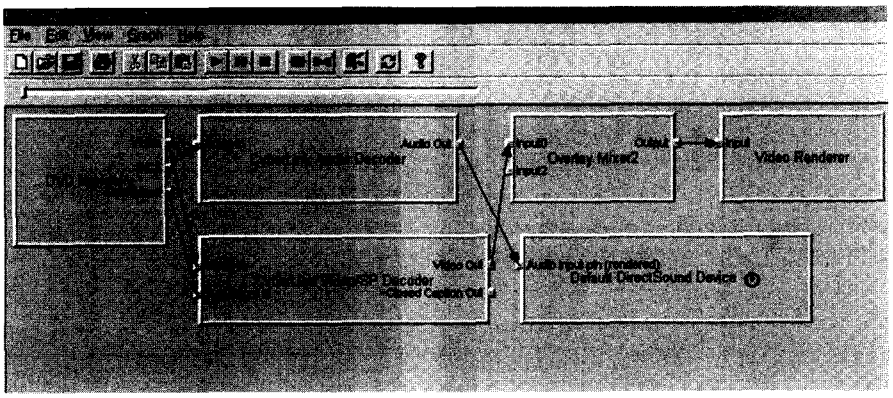

그림 3 DirectShow의 그래프 에디터를 이용하여, DVD 네비게이터를 설계한 예

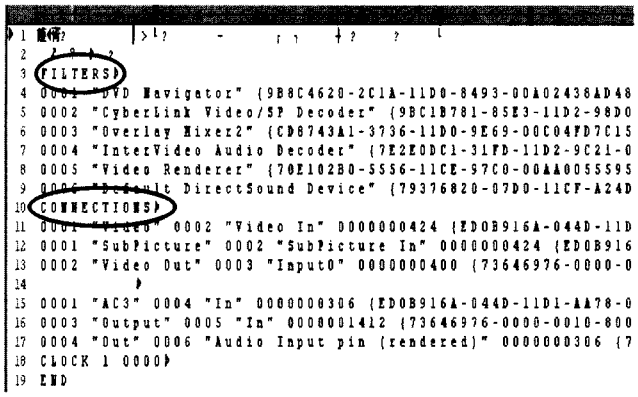

그림 5 GRF 파일의 일부를 텍스트로 변환한 내용

만을 위한 전용 양식으로서 그래프 에디터의 필터 연결 정보를 편집 상태 그대로 완벽하게 저장할 수 있는 장점이 있다. 따라서 본 논문에서는 우선적으로 GRF 파일을 소스 코드 자동 생성의 표준 입력으로 사용하였다.

그림 5는 그림 3의 필터 연결 그래프 예에 대한 GRF 파일의 일부를 편의상 텍스트 파일로 변환한 내용이다. 이 파일은 크게 FILTERS 영역과 CONNECTIONS 영역으로 나뉘는데, 앞부분인 FILTERS 영역에는 사용된 필터들에 대한 정보로서, 각 필터의 명칭과 고유번호(CLSID)를 가지고 있으며, 뒷부분인 CONNECTIONS 영역에는 필터간의 연결 정보로서 입출력 핀의 명칭과 형식, 그리고 연결된 필터들에 대한 정보 등이 기록되어 있다.

3. 소스 코드 자동 생성 시스템

3.1 전체 구성

본 논문에서 제안하는 C 소스 코드 자동 생성 시스템은 그림 2에 나타난 바와 같이 GRF 파일에 저장된 필터 연결 그래프로부터 C 소스 코드를 자동 생성하여 출력하는 단계와 이러한 소스 코드를 직접 호출하여 실행하기 위한 미디어 제어용 소스 코드를 생성하는 단계로 구성되어 있다. 전체적인 소스 코드 생성 과정은 마치 그래프 에디터에서 필터들이 하나씩 선택되어 삽입되고, 또한 각 필터들이 서로 연결되는 과정, 즉 필터 연결 그래프가 생성되는 과정과 매우 유사하게 진행된다.

그림 6은 본 시스템이 C 소스 코드를 자동 생성하는 전체적인 과정을 보여준다. 전체 5개의 단계 중 단계 1은 입력 파일인 GRF 파일로부터 소스 코드 생성에 필요한 각종 정보를 추출하는 과정이며, 단계 2에서 단계

1. 입력파일인 GRF 파일로부터 필터연결그래프 정보를 추출한다.
- 1.1 GRF 파일의 FILTERS 영역에서 사용된 각 필터의 고유번호 및 명칭을 추출한다.
- 1.2 GRF 파일의 CONNECTIONS 영역에서 입출력 핀 및 필터연결정보를 추출한다.

2. 전체 필터연결그래프를 관리하는 필터그래프매니저(헤드 포인터)를 생성하는 소스 코드를 출력한다.
(CoCreateInstance() 이용)

3. 각 필터에 대한 소스 코드를 출력한다.
- 3.1 각 필터에 대한 포인터를 정의/선언하는 코드를 출력한다.
- 3.2 각 필터에 대한 필터 인스턴스를 생성하는 코드를 출력한다. (CoCreateInstance() 이용)
- 3.3 생성된 각 필터 인스턴스를 필터그래프매니저에 등록하는 코드를 출력한다. (AddFilter() 이용)

4. 필터간의 입출력 핀 연결 각각에 대한 소스 코드를 출력한다.
- 4.1 핀 연결을 전체 필터그래프매니저에 등록하는 코드를 출력한다. (ConnectFilters() 이용)

5. 미디어 스트림 제어용 소스 코드를 출력한다.
- 5.1 필터연결그래프에 사용된 소스 필터들에 대해, 적용 가능한 미디어 제어 기능들을 모두 추출한다.
- 5.2 추출된 기능들 중 개발자가 원하는 기능들을 선택하도록 하는 기능 선택 창을 제공한다.
- 5.3 최종 선택된 미디어 제어 기능에 해당하는 미디어 제어용 소스 코드를 출력한다.

그림 6 소스 코드 자동 생성의 전체적인 과정

5는 필요한 소스 코드를 각각 출력하는 과정이다.

그림 7은 그림 6의 자동 생성 과정을 통해 출력된 최종 소스 코드의 주요 내용을 발췌한 것이다. 단계 2에 의해 출력된 첫 번째 문장은 전체 필터 연결 그래프를 관리하는 필터 그래프 매니저를 생성하는 문장으로서 CoCreateInstance() 함수를 이용하여 전체 그래프에 대

```
/* 단계 2: 전체 필터 그래프를 관리하는 필터 그래프 매니저를 생성하는 부분 */
CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER, IID_IGraphBuilder,
        (void **)&m_pGB) ;

/* 단계 3: 주어진 필터 연결 그래프의 각 필터에 대한 소스 코드 부분 */
IBaseFilter *pFilterDVDNavigator;
CoCreateInstance(CLSID_DVDNavigator, NULL, CLSCTX_INPROC_SERVER, IID_IBaseFilter,
        (void **)&pFilterDVDNavigator) ;
m_pGB->AddFilter(pFilterDVDNavigator, L"DVD Navigator");

IBaseFilter *pFilterCyberLinkVideoSPDecoder;
CoCreateInstance(CLSID_CyberLinkVideoSPDecoder, NULL, CLSCTX_INPROC_SERVER,
        IID_IBaseFilter, (void **) &pFilterCyberLinkVideoSPDecoder) ;
m_pGB->AddFilter(pFilterCyberLinkVideoSPDecoder, L"CyberLink Video/SP Decoder") ;

..... // 중략

/* 단계 4: 주어진 필터 연결 그래프의 핀 연결 정보 각각에 대한 소스 코드 부분 */
ConnectFilters(m_pGB, pFilterDVDNavigator, pFilterCyberLinkAudioDecoder) ;
ConnectFilters(m_pGB, pFilterDVDNavigator, pFilterCyberLinkVideoSPDecoder) ;
ConnectFilters(m_pGB, pFilterDVDNavigator, pFilterCyberLinkVideoSPDecoder) ;
ConnectFilters(m_pGB, pFilterCyberLinkVideoSPDecoder, pFilterOverlayMixer) ;
ConnectFilters(m_pGB, pFilterCyberLinkAudioDecoder, pFilterDirectSoundAudioRenderer) ;
ConnectFilters(m_pGB, pFilterOverlayMixer, pFilterVideoRenderer) ;

..... //중략

/* 단계 5: 미디어 제어용 소스 코드 부분 */
IVideoWindow * pVW ;
m_pGB->QueryInterface(IID_IVideoWindow, (void **)&pVW) ;

..... // 중략

HRESULT hr = S_OK;
switch (m_eState)
{
    case Uninitialized:
            return E_FAIL;
    case Play:
            hr = m_pIMC->Run();
            m_pIDvdC2->PlayForwards(1.0, 0, NULL);
            if (SUCCEEDED(hr)) SetState(Playing);
            return hr;
    case Playing:
            if (true == m_bStillOn) {
                hr = m_pIDvdC2->StillOff();
                if (SUCCEEDED(hr)) m_bStillOn = false;
            }
            return hr;
    case Stop:
            hr = m_pIDvdC2->SetOption(DVD_ResetOnStop, TRUE);
            hr = m_pIMC->Stop();
            hr = m_pIDvdC2->SetOption(DVD_ResetOnStop, FALSE);
            break;
    case Paused:
            if (FAILED(m_pIDvdC2->Pause(false)))
                    return false;
            else {
                SetState(Playing);
                return true;
            }
}
if (Playing == m_eState)
{
    if (FAILED(m_pIDvdC2->PlayForwards(2.0, 0, NULL)))
        return false;
    else if (FAILED(m_pIDvdC2->PlayBackwards(2.0, 0, NULL)))
        return false;
    SetState(Scanning);
    return true;
}
```

그림 7 그림 3의 DVD 네이게이터 예제에 대해 자동 생성된 C 소스 코드의 일부 내용

한 인스턴스(즉, 필터 그래프 매니저)를 생성하고, 그 인스턴스를 헤드 포인터 m_pGB가 가리키도록 연결한다. 그 다음 부분은 단계 3의 출력 결과로서, 주어진 필터 연결 그래프의 각종 필터에 대해, 그 필터에 대한 포인터를 선언하는 문장(단계 3.1)과 CoCreateInstance() 함수를 이용하여 해당 필터의 필터 인스턴스 생성하는 문장(단계 3.2), 그리고 AddFilter() 함수를 이용하여, 생성된 필터 인스턴스를 전체 그래프 인스턴스(필터 그래프 매니저)에 연결하는 문장(단계 3.3)이 반복되어 나온다. 그 다음 부분은 단계 4의 출력 결과로서 각 필터 연결에 대해 ConnectFilters() 함수를 이용하여 두 입출력 필터들의 포인터를 상호 연결시키는 문장이 반복되어 나온다. 그리고 나머지 부분은 단계 5의 출력 결과로서 앞의 소스 코드를 호출하여 실행하는 미디어 제어용 함수들이 추가된 것이다.

3.2 생성 시스템의 각 단계별 작업

(가) 단계 1 : GRF 파일 분석 및 필터 연결 그래프의 추출

우선, 첫 번째 과정인 단계 1에서는 필터 연결 그래프 정보가 저장된 GRF 파일이 입력으로 주어지면, 이로부터 필터 연결 그래프를 추출하여, 이 그래프에 사용된 필터와 필터의 입출력 핀, 그리고 필터 연결 정보 등 소스 코드 자동 생성에 필요한 각종 정보를 추출한다.

(나) 단계 2 : 필터 그래프 매니저의 생성

단계 2에서는 전체 필터 연결 그래프를 관리하는 '필터 그래프 매니저' (Filter Graph Manager)를 생성하는 소스 코드를 출력한다. 필터 그래프 매니저는 각 필터들이 제대로 연결되었는지를 감독하고, 미디어 스트림 데이타가 올바른 순서로 필터 사이에 전달되도록 제어하는 DirectShow용 컴포넌트의 한 인스턴스(instance)로서, 이는 마치 전체 필터 연결 그래프에 대한 정보를 가리키는 헤드 포인터(head pointer)의 역할을 한다. 다음 두 단계에 걸쳐, 필터 연결 그래프의 모든 필터와 필터 연결 정보는 이 헤드 포인터를 중심으로 연결 구성된다.

(다) 단계 3 : 필터 연결 그래프의 각 필터에 대한 소스코드 생성

단계 3에서는 각 필터에 대한 소스 코드 생성한다. 필터 연결 그래프에 사용되는 필터들은 모두 DirectShow에서 제공되는 컴포넌트들이다. 따라서 본 단계에서는 필터 연결 그래프에 사용된 각 필터에 대한 포인터를 정의/선언하는 문장과 그 포인터가 가리킬 필터 인스턴스를 새로이 생성하는 문장, 그리고 생성된 필터 인스턴스를 전체 필터 그래프 매니저에 추가 등록하는 문장이 각 필터에 대해 반복되어 출력된다. 그림 7에 나타난 바와 같이, 필터 인스턴스 생성을 위해서는 DirectShow 함수인 CoCreateInstance()가 사용되며, 이렇게 생성된 필터 인스턴스를 필터 그래프 매니저에 추가 등록하기 위해서는 AddFilter()라는 DirectShow 함수가 사용된다.

(라) 단계 4 : 필터 연결 정보에 대한 소스코드 생성

필터간의 입출력 핀 연결 정보에 대한 소스 코드 생성을 하는 단계 4 과정에서는 필터 입력 그래프에 정의된 필터 입출력 핀 연결에 대한 소스 코드를 출력한다. 이를 위해, 연결 정보 각각에 대해, ConnectFilters()라는 DirectShow 함수를 이용하여 입출력 핀으로 연결된 두 필터들의 포인터들을 상호 연결하여 이를 전체 필터 그래프 매니저에게 추가 등록하는 문장을 소스 코드로 출력한다. 이 때 서로 연결되는 필터의 두 입출력 핀들은 핀의 형식이 정확히 일치해야 한다. 만일, 일치하지 않을 경우, ConnectFilters() 함수는 오류를 발생한다. 하지만, 본 연구에서는 그래프 에디터를 통해 실행 결과가 확인된 GRF 파일로부터 필터 및 입출력 핀 연결 정보를 자동 추출하므로, 그림 7에 나타난 바와 같이, ConnectFilters() 함수를 통해 입출력 핀들을 단순히 연결하는 과정만 수행한다.

(마) 단계 5 : 미디어 제어 기능용 함수 코드의 추가

단계 5에서는 다양한 미디어 제어 기능을 지원하기 위해, 적용 가능한 미디어 제어 기능을 모두 추출하여 사용자에게 제시하여, 이중 응용 프로그램에 실제 적용할 기능들을 선택하게 한 후, 선택된 미디어 제어 기능을 수행하는 소스 코드를 출력한다. 입력 파일로부터 추출한 필터 연결 그래프에 대한 소스 코드 생성이 끝나면, 후속 작업으로 이러한 함수들을 호출하여 실행할 미디어 제어 함수들을 소스 코드에 추가하는 작업이 필요하다. 즉, 멀티미디어 응용 프로그램의 사용자는 미디어 제어 함수를 통해 해당 프로그램의 실행을 제어하게 되므로, 개발된 프로그램의 사용자 인터페이스를 추가적으로 구성해 주어야 한다. 이 때, 그 프로그램에서 사용 가능한 미디어 제어 함수는 주어진 필터 연결 그래프에 사용된 필터의 종류 및 속성 등에 따라 달라진다. 이 단계는 기존의 연구와 차별이 되는 부분으로 다음 절에 보다 자세히 설명한다.

3.3 제어 기능 함수 코드 생성

본 연구의 주요 비교 대상인 위저드 시스템[7]의 경우, 필터 연결 그래프로부터 C 소스 코드를 자동 생성하기는 하지만, 멀티미디어 응용 프로그램에서 거의 공통적으로 쓰이는 Play, Stop, Pause 정도의 고정된 미디어 제어 기능만을 제공하며, 이를 자동적으로 추가하거나 수정할 방법은 제공되지 않는다. 따라서 Step Forward, Step Backward, Fast Forward, Rewind 등을 포함한 다양한 미디어 제어 기능을 갖춘 응용 프로그램을 개발하기 위해서는, 별도의 프로그래밍을 통해 자동 생성된 소스 코드를 일일이 다시 수정해 주어야

하는 불편함이 있다.

본 논문에서는, 이러한 점을 개선하여 보다 다양한 기능의 멀티미디어 프로그램 개발 환경을 제공하기 위해, 적용 가능한 미디어 제어 기능을 선택하는 창을 별도로 제공하여, 개발될 프로그램의 실행 버튼(또는 메뉴)으로 표현될 미디어 제어 기능을 자유롭게 선택할 수 있는 기능을 시스템 사용자인 개발자에게 제공한다. 이러한 기능을 제공하기 위해서는, 우선 주어진 필터 연결 그래프에 어떤 종류의 필터가 사용되었는지 그리고 그 필터가 어떤 미디어 제어 기능을 지원하는 지를 파악해야 한다. 그런데, 사용되는 필터의 종류는 GRF 파일로부터 자동 추출이 가능하지만, 각 필터가 갖는 미디어 제어 기능 등의 속성은 그래프로부터 자동 추출할 수 없다. 따라서 본 논문에서는 각 필터에 대한 속성을 수작업으로 미리 분석하여 '필터 속성 파일(Filter Property File)'에 저장해 두고, 이를 토대로 개발될 프로그램, 즉 생성된 소스 코드에 어떤 미디어 제어 기능이 적용 가능한 지를 판단한다.

그런데, 본 연구진의 분석에 의하면, 이러한 미디어 제어 기능을 가지는 필터는 매우 다양하지만, 실제로는 미디어 스트림을 직접 받아들이는 소스 필터에서 제공되는 미디어 제어 기능만이 전체 프로그램 실행에 영향을 미친다. 즉, 소스 필터의 미디어 제어 기능만이 전체 프로그램의 미디어 제어 기능으로 활용될 수 있다. 따라서 본 논문에서는 전체 필터 중 소스 필터에 대해서만 필터 속성 파일을 구성한다. 그리고 필터 연결 그래프에 사용된 필터 중 필터 속성 파일에 정의되지 않는 필터는 소스 필터가 아니므로 이들에 대한 미디어 제어 기능은 아예 추출하지 않고, 소스 필터에 대해서만 정보를 추출하여, 기능 선택 창에 보여준다. 이러한 작업은 속도 개선뿐만 아니라, 미디어 제어 기능을 선택할 개발자로 하여금 기능 선택 과정에서 불필요한 작업을 하지 않도록 도와주는 효과도 있다.

그림 8은 여러 소스 필터의 속성을 정의한 필터 속성 파일의 내용을 보여준다. DirectShow[2]에는 DVD 네비게이터, 오디오 디코더, 비디오 렌더러, 사운드 디바이스, 오버레이 믹서 등 약 68개의 필터가 제공되는데, 이 중 9개 정도가 소스 필터이다. 따라서 이들 소스 필터에 대한 속성, 즉 각 필터의 고유 번호(CLSID) 및 그 필터가 지원하는 미디어 제어 기능 목록을 미리 추출하여 파일 형태로 저장해 둔다. 예를 들어, DVD 네비게이터 필터는 소스 필터로서 고유 번호는 "9B8C4620-2C1A-11D0-8493-00A02438AD48"이며, Play, Pause, Stop, FastForward, Rewind, FullScreen, TitleMenu, Root

```
//DVD Navigator
{9B8C4620-2C1A-11D0-8493-00A02438AD48}
DEFINE_FUNCTIONS{CLSID_DVDNavigator, Play, Pause, Stop, FastFoward, Rewind, FullScreen,
Titlemenu, Rootmenu}

// AVI/WAV File Source
{1643E180-90F5-11CE-97D5-00AA--55595A}
DEFINE_FUNCTIONS{CLSID_AVIWAVFILESOURCE, Play, Pause, Stop, Open, Close, FastFoward,
Rewind, Next, Previous, Zoom}

//Cutlist File Source
{A5EA8D20-253D-11D1-G-B3F1-00AA003761C5}
DEFINE_FUNCTIONS{CLSID_CutlistFileSource, Play, Pause, Stop, Open, Close, FastFoward, Rewind,
StepForward, StepBackward}

//CyberLink DxVA Filter 2
{C494A68B-A398-4E2B-A63A-02578A84DFF4}
DEFINE_FUNCTIONS{CLSID_CyberLinkDxVAFilter2, Play, Pause, Stop}

//File Source(Async.)
{E436EBB5-524F--11CE-9F53-0020AF0BA770}
DEFINE_FUNCTIONS{CLSID_FileSourceAsync, Play, Pause, Stop, Open, Close, FastFoward, Rewind}

//File Source(Netshow URL)
{4B428940-263C-11D1-A520-000000000000}
DEFINE_FUNCTIONS{CLSID_FileSourceNetshowURL, Play, Pause, Stop, Open, Close, FastFoward,
Rewind, Next, Previous}

//File Source(URL)
{E436EBB6-524F-11CE-9F53-0020AF0BA770}
DEFINE_FUNCTIONS{CLSID_FileSourceURL, Play, Pause, Stop, Open, Close, FastForward, Rewind,
Next, Previous, Caption, Subscript}

//File Writer
{8596E5F0-0DA5-11D0-BD21-00A0C911CE86}
DEFINE_FUNCTIONS{CLSID_Filewriter, Play, Pause, Stop, Open, Close}

//IVF Source Filter
{C69E8F40-D5C8-11D0-A520-145405C10000}
DEFINE_FUNCTIONS{CLSID_IVFsourcefilter, Play, Pause, Stop, FastFoward, Rewind}
```

그림 8 필터 속성 파일의 내용

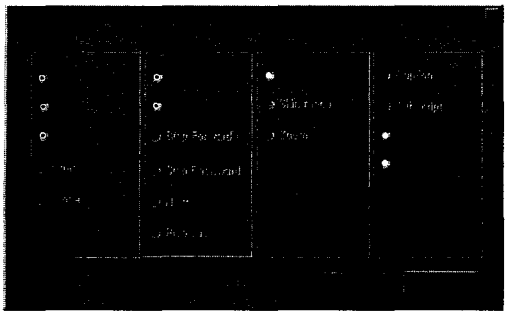

그림 9 응용 프로그램에 적용 가능한 미디어 제어 기능을 선택하는 화면 예

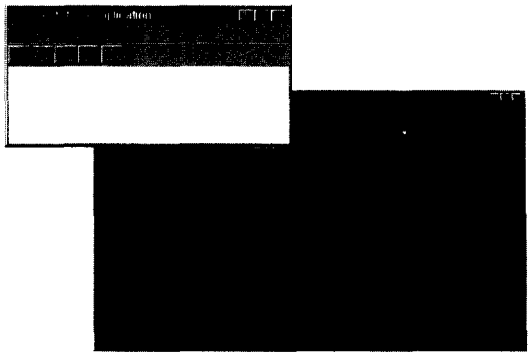

그림 10 자동 생성된 프로그램을 실행시킨 결과 화면 예

Menu 등의 제어 기능을 가질 수 있음을 나타낸다.

위의 과정을 통해, 필터 연결 그래프에 사용된 필터들 가운데 필터 속성 파일에 정의된 소스 필터에 해당하는 필터들이 어떤 미디어 제어 기능을 갖는 지를 자동 추출하였으면, 이를 기능 선택 창을 통해 개발자에게 모두 제시하고, 이 가운데 개발할 프로그램에 최종적으로 추가할 미디어 제어 기능을 자유롭게 선택하도록 한다.

그림 9는 그림 3의 DVD 네비게이터 예제의 경우, 개발자에게 제시되는 기능 선택 창으로서 그림 8의 필터 속성 파일을 토대로 자동 추출된 미디어 제어 기능들을 모두 나열하여 제시하고, 개발자로 하여금 원하는 기능들을 자유롭게 선택하도록 한다. 화면에는 활성화된 기능과 비활성화된 기능이 나열되는데, 활성화된 기능은 그림 3의 예에서 사용된 소스 필터들이 지원하는 기능들이며, 비활성화된 기능들은 필터 속성 파일에는 정의되었으나, 그 기능을 사용하는 소스 필터가 주어진 필터 연결 그래프에는 사용되지 않았음을 의미한다. 그림 9의 경우, DVD 네비게이터 예제에 대해, Play, Pause, Stop, Fast Forward, Rewind, Full Screen, Title Menu, Root Menu 등의 제어 기능만이 활성화되어 있으므로 이러한 기능들은 적용 가능하나, 나머지는 이 예에서는 사용할 수 없는 기능임을 의미한다. 이 예의 경우, 적용 가능한 기능 중 Play, Pause, Stop, Fast Forward, Rewind 기능만이 개발자에 의해 선택되었음을 보여준다.

이런 방식으로, 개발할 응용 프로그램에 적용할 미디어 제어 기능이 최종 선택되면, 선택된 각 미디어 제어 기능을 수행할 미디어 제어용 소스 코드를 전체 소스 코드에 각각 추가해 주어야 한다. 이를 위해, 시스템 내부적으로 개발자가 선택 가능한 모든 미디어 제어 기능, 즉 그림 9에 나열된 모든 기능(활성화된 기능 및 비활성화된 기능) 각각에 대해 추가할 소스 코드를 미리 생성해 두고 있다가, 그 기능이 선택되면, 그 기능에 해당하는 소스 코드를 자동 출력하여 전체 소스 코드에 추가한다.

그림 7의 소스 코드에서 뒷부분은 이런 과정을 거쳐 출력된 미디어 제어용 소스 코드 내용을 보여준다. 그리고 그림 10은 그림 7과 같이 자동 생성된 소스 프로그램을 컴파일하여 실행시킨 경우의 프로그램 실행 결과를 보여준다. 그림 9에서 선택된 미디어 제어 기능 모두가 각각 실행 버튼으로 활성화되어 있음을 알 수 있다.

4. 구현 및 실험 결과 분석

본 논문에서 제안한 시스템은 PC상의 Window 2000 운영체제상에서 C++ 언어(실행파일 545KB)로 구현되었다. 구현된 시스템을 그림 3의 DVD 네비게이터 예제에 적용한 결과, 그림 10에 나타난 바와 같이 수작업을 통해 구현한 결과[11]와 동일한 실행 화면/결과를 보였으며, 소스 코드의 기능적인 면에서도 거의 동일한 결과를 보였다. 여기서 기존 방식의 소스 코드[11]는 본 연구진이 수행한 바 있는 산학연 연구과제에서 개발한 "플랫폼 적응형 DVD Navigator 시스템"의 소스 코드로서, 본 연구와의 객관적인 비교를 위해 그림 9의 미디어 제어 기능과 연관이 없는 부분을 모두 삭제한 나머지 소스 코드를 의미한다.

그림 11은 기존의 수작업을 통한 소스 코드(일부)와 본 시스템에 의해 생성된 소스 코드(일부)를 보여준다. 기존 방식에 의한 소스 코드의 경우, 비교의 편의를 돕기 위해 일부 코드의 순서를 의도적으로 바꾸었으며, 연관이 없는 다른 부분은 생략하였다. 그림에서 보듯이 서로 유사한 기능의 소스 코드가 생성됨을 알 수 있다. 실행 파일 크기의 경우, 기존 방식에 의한 소스 코드를 컴파일할 경우, 226 KB였으나, 자동 생성 기법에 의해 자동 생성된 소스 코드의 경우에는 실행 파일의 크기가 252KB로 약 11% 증가함을 알 수 있었다. 기존의 구현 방식은 다양한 메뉴 및 기능을 자유롭게 추가할 수 있

```
..... 중략 .....

case Graph_Play:
      hr = m_pIMC->Run();
      m_pIDvdC2->PlayForwards(1.0,0,NULL);
      return hr;


case Graph_Stopped1:
      hr = m_pIDvdC2->SetOption(DVD_ResetOnStop, TRUE);
      hr = m_pIMC->Stop();
      hr = m_pIDvdC2->SetOption(DVD_ResetOnStop, FALSE);
      SetState(Graph_Stopped2);
      break;

case Graph_Paused:
      hr = m_pIMC->Pause();
      break;




case Playing:
      if (true == m_bStillOn)
      {
         hr = m_pIDvdC2->StillOff();
         if (SUCCEEDED(hr))
            m_bStillOn = false;
      }
      return hr;
```

<수작업에 의해 구현된 소스 코드>

```
..... 중략.....

  case Play:
       hr = m_pIMC->Run();
       m_pIDvdC2->PlayForwards(1.0, 0, NULL);
       if (SUCCEEDED(hr)) SetState(Playing);
       return hr;

  case Stop:
       hr= m_pIDvdC2->SetOption(DVD_ResetOnStop, TRUE);
       hr = m_pIMC->Stop();
       hr = m_pIDvdC2->SetOption(DVD_ResetOnStop, FALSE);
       break;


  case Paused:
       if (FAILED(m_pIDvdC2->Pause(false)))
            return false;
       else {
          SetState(Playing);
          return true;
       }

 case Playing:
       if (true == m_bStillOn) {
            hr = m_pIDvdC2->StillOff();
          if (SUCCEEDED(hr)) m_bStillOn = false;
       }
        return hr;

 .....

if (Playing == m_eState)
{
     if (FAILED(m_pIDvdC2->PlayForwards(2.0, 0, NULL)))
          return false;
     else if (FAILED(m_pIDvdC2->PlayBackwards(2.0, 0, NULL)))
          return false;
     SetState(Scanning);
     return true;
}
```

<자동 생성 시스템에 의해 생성된 소스 코드>

그림 11 프로그래머가 구현한 소스코드와 본 연구에서 자동 생성된 소스코드의 비교

으며, 프로그래머의 능력에 따라 매우 효과적인 소스 코드를 개발할 수 있는 장점이 있으나, 개발 기간이 많이 소요되는 문제가 있는 반면, 새로운 방식에서는 개발자가 소스 코드를 전혀 수정할 필요가 없어, 코드 작성시 동반될 수 있는 오류를 줄일 수 있으며, 그래프 에디터에서 작성한 필터 연결 그래프로부터 직접 자신이 원하는 C 소스 프로그램을 훨씬 빠른 시간 내에 직접 얻을 수 있기 때문에, 기존 방식에 비해 보다 효과적인 개발이 가능하다.

5. 결 론

본 논문에서는 DirectShow 프로그래밍의 주요 개발 도구인 그래프 에디터의 저장 파일인 GRF 파일로부터 C 소스 프로그램을 자동 생성하는 시스템을 개발하였다. 기존의 DirectShow 프로그래밍 환경에서는 그래프 에디터를 이용한 설계 및 간단한 동작 확인 과정과 응용 프로그램의 구현을 위한 소스 코드 작성 과정이 별도로 이루어진다. 이에 반해, 본 시스템을 사용할 경우, 멀티미디어 응용 프로그램 개발자는 소스 코드를 일일이 작성하거나 수정할 필요 없이, 그래프 에디터를 이용하여 필터 삽입 및 필터 연결 과정을 통해 원하는 프로그램을 작성하고 실행이 잘 되는 지 동작 확인만 한 후, GRF 파일로 저장하기만 하면, 자신이 원하는 C 소스 프로그램을 자동적으로 얻을 수 있기 때문에 보다 효과적인 DirectShow 프로그래밍이 가능하다.

이를 위해, 본 시스템에서는 주어진 필터 연결 그래프의 각 필터에 대해, 그 필터에 대한 포인터를 선언하는 문장과 그 필터에 해당하는 DirectShow 컴포넌트를 찾아 인스턴스를 생성하는 문장, 그리고 그 인스턴스를 전체 그래프 관리 매니저에 등록하는 문장을 소스 코드로 출력한다. 또한 필터 연결 그래프에 정의된 각 필터 연결에 대해, 연결된 두 개의 입출력 필터들을 상호 연결시켜 전체 그래프 관리 매니저에게 등록하는 문장을 소스 코드로 출력한다.

그리고 고정된 개수의 제한된 미디어 제어 기능만을 제공하는 기존의 위저드 시스템과는 달리, 본 시스템은 시스템 사용자인 프로그램 개발자로 하여금 자신이 개발하고자 하는 응용 프로그램에 추가할 미디어 제어 기능을 보다 쉽고 다양하고 선택할 수 있는 기능을 제공한다. 따라서 본 시스템은 DirectShow 프로그래밍을 위한 보다 실용적인 도구로 활용될 수 있다.

하지만, 본 시스템은 현재 DirectShow에서 기본적으로 제공하는 68개의 필터들을 기준으로 개발되었다. 따라서 추가 설치되는 멀티미디어 장비용 디바이스 드라이버나 기타 프로그램 등의 설치로 인해 추가되는 필터, 특히 소스 필터가 있을 경우, 현 시스템이 이를 완벽히 지원하지 못 할 수도 있다. 이를 해결하기 위해서는 추가된 소스 필터에 대한 종류 및 속성 정보를 필터 속성 파일에 수시로 추가하는 방안이 있을 수 있다. 그러나 궁극적으로는 필터 속성 파일을 미리 구성해 두는 대신 시스템으로부터 최신 소스 필터 목록 및 속성 정보를 동적으로 자동 추출하도록 개선하는 것이 필요하다.

참 고 문 헌

[1] DirectX 9.0, http://www.microsoft.com/korea/directx.

[2] DirectX SDK 8.1 Documentation, http://www.DirectX.com.

[3] 이봉하, 윤교철, 김영만, "MPEG 4를 이용한 다자간 멀티미디어 채팅 프로그램 설계 및 구현", 정보과학회 2001년 추계학술대회, 제28권, 제2호, pp.766~768, 2001.

[4] 김운용, 최영근, "디자인 패턴에 대한 소스코드 자동 생성 기법", 한국정보처리학회 논문지, 제9-D권, 제5호, pp.847~858, 2002.

[5] F. Budinsky, M. Finnie, J. Vlissides, and P. Yu, "Automatic Code Generation from Design Patterns", IBM Systems Journal, 35(2), 1996.

[6] 배상현, 남영광, 신규상, "CASE 자료 형식으로부터 CDIF 형식으로 변환하는 프로그램 자동 생성기의 구현", 한국정보처리학회 논문지, 제7권 제12호, pp.3840~3847, 2000.

[7] 차정은, 양영종, 신석규, "EJB 컴포넌트의 코드 자동 생성 도구의 개발", 한국정보처리학회 2001년 추계학술대회, 제8권, 제2호, pp.331~347, 2001.

[8] 신화선, DirectShow 멀티미디어 프로그래밍, 한빛미디어, 서울, 2002.

[9] 조성대, 박우전, "XML을 이용한 JAVA 기반 메뉴 자동 생성 시스템", 정보과학회 추계학술대회, 제27권, 제2호, pp.335~337, 2000.

[10] Brad Vander Zanden, Brad A. Myers, "Automatic, look-and-feel independent dialog creation for graphical user interfaces," CHI: Conference on Human Factors and Computing Systems, pp.27~34, 1990.

[11] 엄성용, 박선화, 차지은, 동지연, "플랫폼 적응형 DVD Navigator의 개발", 한국산학연논문지, 제3권 제2호, pp.107~113, 2003년 6월.

동 지 연

2002년 서울여자대학교 컴퓨터학과 졸업(학사). 2004년 서울여자대학교 대학원 컴퓨터학과 졸업(석사). 관심분야는 영상처리, 컴퓨터 그래픽스

박 선 화

1998년 서울여자대학교 컴퓨터학과 졸업(학사). 2000년 서울여자대학교 대학원 컴퓨터학과 졸업(석사). 2000년~2001년 서울여자대학교 컴퓨터학과 인턴연구원 2001년~현재 서울여자대학교 대학원 컴퓨터학과 박사과정. 관심분야는 CAD 소프트웨어, 모바일 컴퓨팅, 임베디드 시스템

엄 성 용

1985년 서울대학교 컴퓨터공학과 졸업(학사). 1987년 서울대학교 대학원 컴퓨터공학과 졸업(석사). 1992년 서울대학교 대학원 컴퓨터공학과 졸업(박사). 1992년~1993년 컴퓨터신기술공동연구소 특별연구원. 1993년~1995년 University of California, Irvine에서 Post-Doc. 1996년~현재 서울여자대학교 정보통신공학부 부교수. 관심분야는 컴퓨터그래픽스, CAD 소프트웨어, 홈네트워킹, 임베디드시스템